9-1-2015

Dynamic Generalized Extreme Value via Particle Filters

Yonghua Wei

Yonghua Wei

Candidate

Mathematics and Statistics

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Prof. Gabriel Huerta, Chair

Prof. Marios S. Pattichis, Member

Prof. Erik Erhardt, Member

Prof. James Degnan, Member

Dynamic Generalized Extreme Value via Particle Filters

by

Yonghua Wei

B.A., Electrical Engineering, University of New Brunswick, 2004
M.S., Statistics, University of New Mexico, 2014

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Statistics

The University of New Mexico

Albuquerque, New Mexico

July, 2015

Dedication

To my loving parents, Zizhong Wei and Ailan Wang, who guided me to where I am today

To my wife, Yan, for her support, encouragement and the best gift she gives to me

To my daughters, Alisa and Isabella, the best gift EVER

In memory of my grandma, I know you would be so proud

Acknowledgments

I would especially like to thank my advisor, Professor Gabriel Huerta, for his generous guidance and support, for many illuminating discussions over the past two years, and for his sharp comments and enormous help on the manuscript.
I would also like to thank Professor Jonathan Stroud from George Washington University for his valuable inputs.
I wish to express my sincere gratitude to the members of my committee: Professor Marios S. Pattichis from the Computer Engineering department, Professor Erik Erhardt and Professor James Degnan from the Department of Mathematics and Statistics for their suggestions that led me to improve this work. I have truly enjoyed interacting with each one of them.

Dynamic Generalized Extreme Value via Particle Filters

by

Yonghua Wei

B.A., Electrical Engineering, University of New Brunswick, 2004
M.S., Statistics, University of New Mexico, 2014

Ph.D., Statistics, University of New Mexico, 2015

Abstract

This thesis considers an introduction of recently developed Particle Filter algorithms and their application to Dynamic Linear models. We start from the case of a dynamic model without fixed parameter estimation to demonstrate how particle filter work. Then we show how advanced particle filters work on posterior density approximations for models that allow dynamic states and fixed parameters estimation simultaneously. It is proven that these state and parameter estimations can be achieved for these classes of dynamic models via efficient Particle Filter methods without the need of the more traditional Forward Filtering Backward Sampling (FFBS) simulation. Simulations for the first order dynamic linear model and a time-varying extreme value analysis via the Generalized Extreme Value distribution are illustrated using particle filter methods and a MCMC algorithm through sampling of full conditionals that involve a variety of Metropolis-Hastings steps. Additionally, we illustrate all the different methodologies with three practical time

series of extreme values. The athletic records originally analyzed in Robinson & Tawn (1995) and the follow-up discussions in Smith (1997) and Robinson & Tawn (1997), the maximum monthly rainfall values from January 1961 to November 1999 taken at the Maiquetía station near Caracas, Venezuela discussed in Huerta and Sansó (2007) and and Huerta and Stark (2012), the minimum daily stock returns occurring during a month from January 4, 1990 to December 28, 2007 using the Tokyo Stock Price Index (TOPIX) as in Nakajima, Kunihama, Omori and Frühwirth-Schnatter (2011).

Contents

List of Figures

Glossary

y_t	Observation at time t.
x_t	State of the dynamic model at time t, where $t = \{1, 2..T\}$.
T	Total number of observations.
j	Particle index with $j = \{1, 2..N\}$.
N	Total number of particles in the particle filter.
x_t^j	j^{th} Particle of state x_t.
$\overleftarrow{x_t}$	Smoothed state of the dynamic model at time t.
$\overleftarrow{x_t}^j$	j^{th} particle of smoothed state of the dynamic model at time t.
m	MCMC iteration index with $m = \{1, 2..M\}$.
θ	Static parameter vector.
X_i	Multivariate parameter and states combination $X = (X_1, \ldots, X_d)$.
μ_t	Location/state of the GEV distribution at time t, $t = \{1, 2..T\}$.
σ	Scale parameter of the GEV distribution.
ξ	Shape parameter of the GEV distribution.

Chapter 1

Introduction to Dynamic Linear Model

1.1 Dynamic Linear Model

Consider a time series $y_{1:T} = \{y_1, y_2, \ldots, y_{t-1}, y_t, y_{t+1}, \ldots, y_T\}$, where y_t is the observation at time t and T is the total number of observations. For making inference on the time series, in particular for predicting the next time point value y_{t+1} given the previous stage observations $\{y_1, \ldots, y_t\}$, we need to specify the probability law of the process y_t. State space models are based on the idea that the time series $y_{1:t} = \{y_1, y_2, \ldots, y_t\}$ is an incomplete and noisy function of some underlying unobservable process $x_{1:t} = \{x_1, x_2, \ldots, x_t\}$, called the latent states. See for example Petris, et al. (2009), Tanizaki (2004), West and Harrison (1997) and Prado and West (2010).

The relation between the observations $\{y_1, \ldots, y_t\}$ and the unobservable states $\{x_1, x_2, \ldots, x_t\}$ is shown in Figure 1.1.

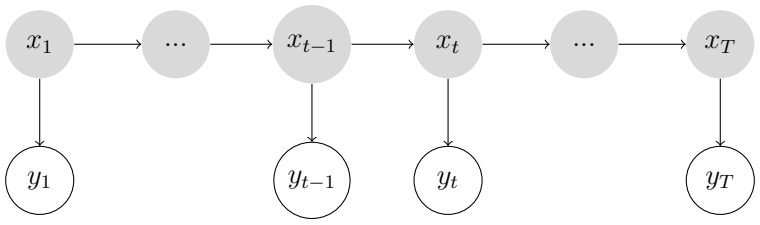

Figure 1.1: Illustration of the relation between observations and states in a dynamic model. y_t: observation; x_t: state.

We can reasonably assume that the observation y_t only depends on x_t, the state of the system at the time the measurement is taken. The observations $y_{1:t}$ are conditionally independent provided that $x_{1:t}$ are known. In other words, each y_t only depends on x_t and therefore,

$$y_t|x_t \sim p_{y|x}(y|x_t) \tag{1.1}$$

where $p_{y|x}(y|x_t)$ is the observation distribution function.
To simplify the analysis, the unobservable state x_t can be modeled as a first order latent or hidden Markov process with an initial distribution $p(x_0)$ and the following evolution process,

$$x_t|x_{t-1} \sim p_X(x_t|x_{t-1}) \qquad t = 1, \ldots, T. \tag{1.2}$$

As discussed by West and Harrrison (1997), $p(x_0)$ contains all the available relevant starting information used to form initial views about the state at time 0, including its history and all defining model quantities.
Dynamic linear models are a broad class of linear state space models with time varying parameters to model time series data (Migon, Gamerman, Lopes and Ferreira 2005). Let y_t be an $(r\times 1)$ vector observation and x_t be an $(n\times 1)$ parameter vector, a basic linear state space model takes the form (West and Harrison 1997):

$$\text{Observation equation:} \qquad y_t = F_t' x_t + v_t, \qquad v_t \sim N(0, V_t). \tag{1.3}$$

$$\text{Evolution equation:} \qquad x_t = G_t x_{t-1} + \omega_t, \qquad \omega_t \sim N(0, W_t). \tag{1.4}$$

for each time t, where

1. F_t is a known $(n \times r)$ matrix;
2. G_t is a known $(n \times n)$ matrix;
3. V_t is a known $(r \times r)$ variance matrix;
4. W_t is a known $(n \times n)$ variance matrix.

The first equation, called observation or measurement equation, describes the relation between the observed time series, y_t, and the unobserved state, x_t. In general, it is assumed that the data y_t are measured with error, which is represented by the measurement error v_t. The second equation, called evolution or transition equation, describes the evolution of the state variables as being driven by the state variable at previous time x_{t-1} and the evolution error ω_t. Both errors are Normally distributed and with variance matrix Σ_v and Σ_ω correspondingly. The choice of F and G depends on the model and the nature of the data that is being analyzed. A much more general form of state space model can be extended as, the system matrices F and G could depend explicitly on time, or one could introduce some other unknown parameters into the model specification (West and Harrrison 1997).

1.2 First Order Dynamic Linear Model

The first order dynamic linear model, or the first order polynomial linear model is the simplest and most widely used dynamic linear model. In this model, the

observation equation takes the form

$$y_t = x_t + v_t, \qquad v_t \sim N(0, V_t) \tag{1.5}$$

where x_t is the mean of the observation at time t and V_t is the observation variance at time t. The time evolution of the mean is modeled as a simple random walk

$$x_t = x_{t-1} + \omega_t, \qquad \omega_t \sim N(0, W_t) \tag{1.6}$$

to represent the evolution equation where W_t is the evolution variance at time t. Normally, the evolution error ω_t is independent of the observation error v_t and the evolution variance W_t is much smaller compared to the observation variance V_t. As mentioned by West and Harrison (1997), this model is used effectively in numerous applications, particularly in short-term forecasting for production planning and stock control. In a product demand forecasting example, x_t represents the true underlying market demand at time t where v_t represents random fluctuations which reflect the discrepancy of individual customer orders.

1.3 General Dynamic Model

Although the first order dynamic linear model is efficient and easy to implement, many real case examples need to include parameter non-linearity into the standard DLM model. In order to model accurately the underlying dynamics of a physical system, it is important to include elements of non-linearity and non-Gaussianity in many application areas. Therefore, the Markovian general dynamic model has the following form:

$$y_t \;=\; f_t(x_t) + v_t, \qquad v_t \sim N(0, V_t) \tag{1.7}$$

$$x_t = g_t(x_{t-1}) + \omega_t, \qquad \omega_t \sim N(0, W_T) \tag{1.8}$$

where f_t is a known non-linear function mapping the state vector x_t to the observation and g_t is a known non-linear vector evolution function. The most known subclass of dynamic models is the *normal dynamic linear model*, referred to simplify as the *dynamic linear model*, or DLMs. The first order dynamic linear model is just a special case of the general dynamic model where f_t and g_t are the identity functions.

1.4 Inference for the Dynamic Model via MCMC

In this section we outline the Kalman Filter analysis on a first order DLM. Next, a MCMC posterior simulation for the general framework is introduced, which is necessary for the class of nonlinear/non-Gaussian dynamic models. We will end this chapter with algorithms specially designed for the Gaussian dynamic linear model.

1.4.1 State Estimation

For a given state space model, the main tasks are to make inference on the unobserved states or predict future observations based on a part of the observation sequence. To estimate the state vector, *filtering* and sometimes *smoothing* density calculations are needed.

In an online setting, such as with target tracking, we are interested in estimating recursively the joint posterior distribution $p(x_{1:t}|y_{1:t})$ or just the marginal posterior density $p(x_t|y_{1:t})$, the density of the current state x_t conditional on a set of sequential observations $y_{1:t}$ up to time t. This density is known as the *filtering*

distribution (Doucet, et al 2001).

While *filtering* corresponds to estimating the distribution of the current state based on all the observations received up to the current time, *smoothing*, or retrospective analysis, corresponds to estimating the distribution of the state at a particular time given all of the observations up to some later time (Doucet and Johansen 2008). Observing the values until the last time point T, we may extrapolate back over time to compute the joint posterior density $p(x_{1:t}|y_{1:T})$ or to approximate the associated marginals for an early time point t: $p(x_t|y_{1:T})$. For example, in the stock market, an analyst might have the time series of the stock price for a certain number of days, she may update the current estimates as new data are observed (*filtering*); or she might be interested in retrospectively understanding the operating behavior for the past (*smoothing*).

1.4.2 Filtering and Smoothing

As discussed by Gordon, et al, (1993), Briers, et al, (2010), Doucet and Johansen (2008) and Sarkka (2013), the filtering density $p(x_t|y_{1:t})$ can be achieved recursively through two steps: one-step ahead prediction and filtering.

(i) Forward Evolution or Prediction. In this step, we compute the current state marginal distribution $p(x_t|y_{1:t-1})$ given the posterior for the last state x_{t-1} and the observations up to the last stage $y_{1:t-1}$. This is also called propagation step and the evolution equation is,

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}. \tag{1.9}$$

(ii) Update or Filtering. We update the filtering density on observing y_t and compute the current posterior with Bayes theorem,

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}). \tag{1.10}$$

1.4.3 Smoothing

Smoothing refers to the backward sampling step and depends on the following fact,

$$p(x_{0:T}|y_{1:T}) \propto p(x_T|y_{1:T}) \prod_{t=1}^{T} p(x_{t-1}|x_{t:T}, y_{1:T}). \tag{1.11}$$

Due to the Markovian property of the evolution function in a general dynamic model, we get

$$\begin{aligned} p(x_t|x_{t+1:T}, y_{1:T}) &= p(x_t|x_{t+1}, y_{1:t}) \\ &= \frac{p(x_{t+1}|x_t)p(x_t|y_{1:t})}{p(x_{t+1}|y_{1:t})} \\ &\propto p(x_t|y_{1:t})p(x_{t+1}|x_t). \end{aligned} \tag{1.12}$$

which shows the smoothing density is proportional to the product of the filtering density $p(x_t|y_{1:t})$ and the evolution density $p(x_{t+1}|x_t)$.
The smoothing sampling begins with time T (i.e., the endpoint case in the smoothing algorithm) is equivalent to the filtering density at time T. It starts from drawing samples $x_T \sim p(x_T|y_{1:T})$ and then recursively by using equation 1.12, to get smoothed samples at $t-1$ by re-sampling the filtering samples with weight proportional to $p(x_{t+1}|x_t)$.

1.4.4 Kalman Filter for Dynamic Linear Models

The previous section discusses the general principles of the Bayesian filtering and smoothing algorithm. However, in general the actual computation of these conditional distributions is complicated. One great achievement in this area is the popular Kalman filter that was proposed in the 1960's: "The Kalman filter (KF) is a closed form solution to the linear Gaussian filtering problem. Due to linear Gaussian model assumptions the posterior distribution is exactly Gaussian and

no numerical approximations are needed." (Sarrka 2013). It is easy to prove that dynamic linear models have a Gaussian distribution for any $t \geq 1$ states as well. Thus the filtering and smoothing densities are completely determined by their means and variances.

In this section we introduce the Kalman Filter through a DLM defined by equations 1.3 and 1.4 with $F_t^{'}, G_t, V_t$ and W_t are completely known and initial information of $p(x_0) = N(m_0, C_0)$. More details can be found in West and Harrison (1997).

Filtering:

(a) Start from the posterior at $t-1$:

$$(x_{t-1}|y_{1:t-1}) \sim N(m_{t-1}, C_{t-1}). \tag{1.13}$$

(b) Get the prior at time t with the evolution equation:

$$(x_t|y_{1:t-1}) \sim N(a_t, R_t), \tag{1.14}$$

where $a_t = G_t m_{t-1}$ and $R_t = G_t C_{t-1} G_t^{'} + W_t$.

(c) The one-step prediction at time $t-1$ from the observation equation:

$$(Y_t|Y_{1:t-1}) \sim N(f_t, Q_t), \tag{1.15}$$

where $f_t = F_t^{'} a_t$ and $Q_t = F_t^{'} R_t F_t + V_t$.

(d) The posterior at time t is computed with Bayes theorem:

$$(x_t|Y_{1:t}) \sim N(m_t, C_t), \tag{1.16}$$

and gives the following recursive equations:

$$\begin{aligned} m_t &= a_t + A_t e_t, & (1.17)\\ C_t &= R_t - A_t Q_t A_t^{'}, & (1.18)\\ A_t &= R_t F_t / Q_t, & (1.19)\\ e_t &= Y_t - f_t. & (1.20) \end{aligned}$$

Smoothing:

To perform smoothing, we need to consider one more term for the filtering steps:

$$B_t = C_t G'_{t+1}/R_{t+1} \tag{1.21}$$

Then for all k, $(1 \leq k < T)$, the marginal smoothing distributions are

$$(x_{T-k}|Y_{1:T}) \sim N(mm_{t-k}, CC_{t-k}) \tag{1.22}$$

where,

$$mm_{t-k} = m_{t-k} + B_{t-k}(mm_{t-k+1} - a_{t-k+1}) \tag{1.23}$$

and

$$CC_{t-k} = C_{t-k} + B_{t-k}(R_{t-k+1} - R_{t-k+1})B'_{t-k} \tag{1.24}$$

with starting values,
$mm_T = m_T$ and $CC_T = C_T$.
Therefore, when a DLM is completely specified, i.e., there are no unknown parameters except the state vector, one can use the well-known Kalman filtering and smoothing algorithms to obtain means and variances of the conditional distributions of the states given the observed data.

1.4.5 MCMC _ Hybrid M-H within Gibbs

We have seen previously that the Kalman Filter can provide an analytic solution of the sequence of densities $p(x_{1:T}|y_{1:T})$ for the Gaussian dynamic linear model given known measurement and evolution variance W_t and V_t. Unfortunately, real-world models are rarely that simple enough to be solved exactly. We often have to estimate unknown parameters, in the context of non-linearity or/and non-Gaussianity.

While for a few very special cases, as seen in West and Harrison (1997), it is possible to compute the posterior distribution of unknown parameters in closed form for many cases, we have to rely on Monte Carlo methods to draw samples from the target posterior distribution.

Markov Chain Monte Carlo (MCMC) methods draw samples by simulating a Markov chain having the target distribution as the stationary distribution. In the case of models that are Gaussian and linear, or conditionally so, drawing states is a natural component of Gibbs sampling methods for learning about the posterior distribution of states, parameters and other functions of interest. Carlin, Polson and Stoffer (1992), and Carter and Kohn (1994) introduced the scheme of sampling nonlinear and/or non-Gaussian state-space models with Gibbs sampling. The transition kernel of Gibbs sampling is formed by the full conditional distributions and Gibbs sampling is generally used only when full conditional distributions are known and easy to sample from. Nevertheless, often it is not possible to simulate from one or more of the full conditional densities such as those cases with densities not in standard form and that have intractable normalizing constants. Then the posterior sampling is usually conducted via other schemes, very often with the Metropolis-Hastings (MH) algorithm, as in Hastings (1970). For each component of the parameter vector, a new sample is drawn from a "proposal" distribution, then decide whether to take the proposed sample or keep the current value using an acceptance probability:

$$\alpha(\theta^\star, \theta) = \min\left\{1, \frac{\pi(\theta^\star)/J(\theta^\star|\theta)}{\pi(\theta)/J(\theta|\theta^\star)}\right\}$$

where $\theta^\star$ is the propose parameter values evolved from current parameter value θ with the proposal/transition kernel $J(\theta^\star|\theta)$ to guide the move. The fraction inside the brackets

$$\frac{\pi(\theta^\star)/J(\theta^\star|\theta)}{\pi(\theta)/J(\theta|\theta^\star)}$$

is called the importance ratio (Gamerman and Lopes 2006).

Denote $X = (X_1, \ldots, X_d)$ a vector that represents a combination of parameters and states and assume the marginal and conditional distributions of the components X_i can be obtained from the joint density $\pi(X_1, \ldots, X_d)$. Let $X_{-i} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_d)$ represent the vector X with its ith component removed. The conditional distribution of $X_i|X_{-i}$, called full conditional distribution of X_i is denoted by $\pi_i(X_i)$. By conditional probability, we have $\pi(X) = \pi_i(X_i)\pi(X_{-i})$. Furthermore, a component proposal/transition kernel $J_i(X_i^\star|X_i)$ is used to guide the move of the ith component of X while the other components of X are kept unchanged, thus

$$\frac{\pi(X_1, \ldots, X_{i-1}, X_i^\star, X_{i+1}, \ldots, X_d)}{\pi(X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_d)} = \frac{\pi_i(X_i^\star)\pi(X_{-i})}{\pi_i(X_i)\pi(X_{-i})} = \frac{\pi_i(X_i^\star)}{\pi_i(X_i)}$$

This provides a reducible Markovian chain to propose a transition on component X_i with kernel J_i while the other components of X remain fixed and are not affected. Thus the importance ratio may be written as:

$$r = \frac{\pi_i(X_i^\star)/J_i(X_i^\star|X_i)}{\pi_i(X_i)/J_i(X_i|X_i^\star)} \tag{1.25}$$

instead of the joint density formula of

$$r = \frac{\pi(X^\star)/J_i(X^\star|X)}{\pi(X)/J_i(X|X^\star)} \tag{1.26}$$

with $X^\star = (X_1, \ldots, X_{i-1}, X_i^\star, X_{i+1}, \ldots, X_d)$ and $X = (X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_d)$.

Geweke and Tanizaki (1999) proposed the nonlinear and non-Gaussian smoother using the hybrid Metropolis-Hastings algorithm within Gibbs sampling. This is also illustrated in Robert and Casella (2004) and Gamerman and Lopes (2006). In this thesis, I apply the hybrid method due to its flexibility of dealing with non-linear and non-Gaussian dynamic models. The component-wise version of the hybrid algorithm suggested in Prado and West (2010) is as follows:

1. Initialize the iteration index $m = 1$ and set the initial value of the chain at $X^{(0)}$.

2. Start from the component index $i = 1$.

3. Propose the ith component of the vector of states of the chain to a new value $X_i^\star$ based on the proposal density J_i.

4. Calculate the importance ratio as in equation (1.25).

5. Set

$$X_i^{(m)} = \begin{cases} X_i^\star & \text{with probability min(r, 1) ,} \\ X_i^{(m-1)} & \text{otherwise .} \end{cases} \tag{1.27}$$

6. Change the component index to the next one $i+1$ and return to step 3 until $i = d$.

7. Change the iteration index to $m + 1$ until convergence is reached.

A critical issue for using the MCMC method in applications is how to determine when it is safe to stop sampling and use the samples to estimate the posterior distribution of interest. The textbook by Christensen, et al. (2011) mentions that the study of convergence in MCMC is usually approached by analyzing the observed output from the sample chain empirically. Initial poor estimates are not uncommon in MCMC implementation since we may not have good knowledge on how to set the initial values. *Burn-in* is frequently used to deal with this issue, which refers to the practice of discarding initial iterations of sampled values. A reasonable burn-in value will be selected so that the sampling density achieves stationarity. Trace plots of each component will be used to check stationarity of the chain by showing there is no trend of pattern after burn-in. A second diagnostics is the lagged autocorrelation function. Large autocorrelations will suggest non-independence of samples which may need a huge number of sampling iterations for MCMC convergence. Usually, even after the burn-in phase, significant autocorrelations in the observed output terms are present especially for complicated

models. To reduce or even remove this autocorrelation effects, we can select samples by reducing the output or sample values by keeping only every k^{th} iteration. This process is known as *thinning*.

Chapter 2

Particle Filter Methods

Particle Filter or *Sequential Monte Carlo (SMC)* methods are a set of online algorithms that estimate the posterior density of the state space vector by directly implementing the Bayesian recursive equations of a state space model. Many real-world data analysis tasks involve observations that arrive sequentially in time and one could be interested in performing Bayesian inference in actual time. Kalman filters were developed to track the state of a dynamic system for which a Bayesian model exists, they provide an exact solution analytically. But very often, that exact solution is computationally expensive or even not achievable.

Whereas for an online setting, one should update the posterior distribution as data become available in order to take full advantage of sequentially arrived data. In cases where the state and observation models are linear and Gaussian, it is possible to evaluate the posterior density $p(x_{1:t}|y_{1:t})$ in closed form using a Kalman filter (Kalman, 1960). Nonetheless, in the case of nonlinear, non-Gaussian state space models, a closed form solution for the posterior density is usually difficult or not available. MCMC schemes can be designed to approximate the posterior distribution, but it is often difficult to make them efficient for computation in high dimensional spaces. This lead to the development of a class of approximation

techniques known as sequential Monte Carlo methods, or particle filters (Nemeth, Fearnhead and Mihaylova 2013).

2.1 Basic Particle Filter

In its most basic form, particle filters work by starting with a sample from the posterior at time $t-1$, predicting the state at time t, and then updating the importance weights based on the observation y_t. In this section, I will introduce the concept of filtering first and later provide the basic particle filter algorithms which focus on state estimation.

2.1.1 Filtering Algorithms

In what can be considered the seminal work in the particle filtering literature, Gordon, Salmond and Smith (1993) developed a strategy based on a sequence of importance sampling steps. Filtering can be thought of as the repeated application of a two-stage procedure. First, the current density must be propagated into the future via the state transition density $p(x_t|x_{t-1})$ to produce the *prediction density*

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}, \tag{2.1}$$

where $p(x_t|y_{1:t-1}) = p(x_t|y_1, \ldots, y_{t-1})$.
Second, one moves to the *filtering density* via Bayes theorem,

$$p(x_t|y_{1:t}) \quad = p(x_t|y_t, y_{1:t-1}) \quad = \frac{p(y_t|x_t)f(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}$$

where in the denominator

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t. \tag{2.2}$$

See Appendix A for a detailed proof.

This implies that the data can be processed in a single sweep, updating our knowledge about the states as we receive more information.

Particle filters are a class of simulation-based filters that recursively approximate the filtering random variable $(x_t|y_{1:t})$ by a set of particles $x_t^1, x_t^2, ..., x_t^N$. Each particle has a weight or discrete probability mass of $\omega_t^1, ..., \omega_t^N$ assigned to it that represents the probability of each particle being sampled from the probability density function. These discrete particles are viewed as samples from $f(x_t|y_{1:t})$ so a continuous pdf is being approximated by these discrete particles. Throughout, N is taken to be very large and we require that as $N \to \infty$, the particles can be used to approximate increasing well the density of $(x_t|y_{1:t})$ as shown in Prado and West (2010):

$$\hat{p}(x_t|y_{1:t}) = \sum_{j=1}^{N} \omega_t^j \delta_{x_t^j}(x_t) \approx p(x_t|y_{1:t}) \tag{2.3}$$

where δ denotes the Dirac delta function. The particles at time $(t-1)$, $\{x_{t-1}^1, x_{t-1}^2, ..., x_{t-1}^N\}$, are updated to obtain a new set of particles at time t via importance sampling and re-sampling. We now discuss various algorithms for updating the particles from time $t-1$ to t.

2.1.2 Sequential Importance Sampling and Re-sampling (SIS and SIR)

The sequential importance sampling (SIS) is a sequential version of importance sampling. The SIS algorithm can be used to approximate the joint density $p(x_{0:t}|y_{1:t})$, here x_0 is used to represent the state at time 0. Refer to Liu and Chen (1998) and Doucet, de Freitas and Gordon (2001) for a more detailed discussion. At time $t-1$, assume that $p(x_{0:t-1}|y_{1:t-1})$ is approximated by the weighted set of particles $\{(x_{0:(t-1)}, \omega_{t-1})^j; j = 1, ..., N\}$. Then for each j, with $j = 1 : N$,

1. Sample x_t^j from an importance or proposal density $g_t(x_t|x_{0:(t-1)}^j, y_{1:t})$ and set $x_{0:t}^j = (x_{0:(t-1)}^j, x_t^j)$.

2. Compute the importance weights ω_t^j as

$$\omega_t^j \propto \omega_{t-1}^j \frac{p(y_t|x_t^j)p(x_t^j|x_{t-1}^j)}{g_t(x_t^j|x_{0:(t-1)}^j, y_{1:t})} \tag{2.4}$$

   see appendix A for why the SIR method works.

Finally, use $\{(x_{0:t}, \omega_t)^j; j = 1, ..., N\}$ as a particle approximation to $p(x_{0:t}|y_{1:t})$. If we continue this process until the last time point T, then we get $\{(x_{0:T}, \omega_T)^j; j = 1, ..., N\}$ as a particle approximation to $p(x_{0:T}|y_{1:T})$, which is the full posterior distribution of $x_{0:T}$, or the joint smoothing density conditional on all observations. This is the reason why the SIS method is called particle smoother in Sarkka (2013). Although the SIS method looks attractive due to its straight forward one step solving for both the filtering and smoothing density, in practice, it is difficult to achieve reasonable samples due to the so called issue of *degeneracy*.

As discussed in Doucet, Godsill and Andrieu (2000), it can be shown that the variance of the importance weights increases over time, leading to a problem known as partial *degeneracy*. That is, after some iterations of the algorithm all except one particle will have weights that are very close to zero. The degeneracy problem can be reduced by using a *re-sampling* procedure. This refers to a procedure where we draw N new samples from the discrete distribution defined by the weights and replace the old set of N samples with this new set.

The re-sampling procedure can be described as follows:

1. Interpret each weight ω_t^j as the probability of obtaining the sample index for the set $\{x_t^j, j = 1, ...N\}$.

2. Draw N samples from this discrete distribution and replace the old sample set with this new one.

3. Set all weights to a constant value so $\omega_t^j = 1/N$.

The idea of the re-sampling procedure is to remove particles with very small weights and duplicate particles with large weights.
Adding a re-sampling step to the sequential importance sampling algorithm leads to the so called sequential importance re-sampling (SIR) (Gordon, et al., 1993; Kitagawa, 1996; Doucet, et al., 2001). In SIR, re-sampling is not usually performed at every time step, but only when it is actually needed. The so-called effective sample size (Liu 1996) given by

$$M_{t,eff} = \frac{1}{\sum_{j=1}^{N}(\omega_t^j)^2}, \tag{2.5}$$

can be used as a measure of degeneracy. Note that $M_{t,eff} = N$, the total number of particles, when $\omega_t^j = 1/N$ for all the particles and $M_{t,eff} = 1$ when a single particle has weight equal to one. A small value of the effective size indicates particle degeneracy. Typically we do resampling only when the effective sample size is below $N/2$, or half of the number of particles. A SIR algorithm based on the effective number of particles is implemented in Prado & West (2010):

1. Sample x_t^j from an importance density $g_t(x_t|x_{0:(t-1)}^j, y_{1:t})$
   and set $x_{0:t}^j = (x_{0:(t-1)}^j, x_t^j)$.

2. Compute the importance weights ω_t^j as

$$\omega_t^j \propto \omega_{t-1}^j \frac{p(y_t|x_t^j)p(x_t^j|x_{t-1}^j)}{g_t(x_t^j|x_{0:(t-1)}^j, y_{1:t})}. \tag{2.6}$$

3. Compute $M_{t,eff}$. If $M_{t,eff} < M_0$ for some specified minimum sample size M_0, usually a percentage of the number of particles, N, then do the re-sampling as follows:

   (a) For $j = 1 : N$ sample $x_{0:t}^j$ with probability ω_t^j from the particle approximation obtained in Step 2.

   (b) Set $\omega_t^j = 1/N$ for all j

   and report the new particle set $\{(x_{0:t}, \omega_t)^j; \omega_t^j = 1/N; j = 1, ..., N\}$ as a particle approximation to $p(x_{0:t}|y_{1:t})$.

We have seen that SIS and SIR can provide an approximation of the smoothing density $p(x_{0:T}|y_{1:T})$ upon arriving to the last time point T. To get the smoothing solution from SIR we need to re-sample the state histories but not only the current states. However, these approximations are poor when T is large because of the degeneracy problem where for some small value t, all $x_t^j, j = 1...N$ are the same particle. An alternative scheme for obtaining the smoothing density will be discussed later.

If the objective is to approximate $p(x_t|y_{1:t})$, the filtering distribution, then the first t components in each path $x_{0:t}^j$ can be discarded as long as the calculation of weights depends only on x_t^j and x_{t-1}^j (Prado and West 2010, Klaas, de Freitas and Doucet 2012, Lindsten, Schn and Svensson 2012, Sarkka 2013). Since we are interested in the filtering distributions only, we discard the sample histories $x_{0:t-1}$ and only keep the current states x_t for later implementations of the SIR algorithm in this dissertation.

2.1.3 Bootstrap Filter

The bootstrap filter (Gordon, et al. 1993) is a variation of the SIR method where the state transition density, or "prior kernel" $p(x_t|x_{t-1})$ is solely used as the importance distribution $g_t(x_t|x^j_{0:(t-1)}, y_{1:t})$. Cappe, Godsill and Moulines (2007) mentioned that a distinctive feature of the bootstrap filter is that the weight does not depend on the past trajectory of the particles but only on the likelihood $p(x_t|x_{t-1})$. The use of the prior kernel is popular because sampling is often straightforward, and computing the incremental weight simply amounts to evaluating the conditional likelihood of the new observation given the updated particle position. Then the importance weight in equation (2.4) is simplified to

$$\omega^j_t \sim \omega^j_{t-1} p(y_t|x^j_t).$$

Bootstrap filter algorithm (Sarkka 2013) :

- Suppose at time $t-1$, $p(x_{t-1}|y_{1:t-1})$ is approximated by $\{(x_{t-1}, \omega_{t-1})^j; j = 1, ..., N\}$.
- Approximate the next stage filtering density $p(x_t|y_{1:t})$:

  For each j,

  1. Sample a new point using the evolution density from the dynamic model:

  $$x^j_t \sim p(x_t|x^j_{t-1}) \tag{2.7}$$

  2. Calculate the weight:

  $$\omega^j_t \propto \omega^j_{t-1} p(y_t|x^j_t) \tag{2.8}$$

  3. Do re-sampling if needed.

In the original version of the bootstrap algorithm, re-sampling is carried out at each and every time step, in which case the term $\omega_{t-1}^j = 1/N$ is a constant and may thus be ignored. With this form of the importance distribution, we do not need to store the whole histories $x_{0:t}^j$ as in the SIS algorithm, only the current states x_t^j. This form is also convenient in other particle filters discussed in the next section, since we do not need to worry about the state histories during the re-sampling step as in the particle smoother (Sarkka 2013).

Despite its appealing properties and ease of implementation, the use of the state transition density as importance distribution can often lead to poor performance and may require a very large number of Monte Carlo samples for accurate estimation due to the inefficiency of the importance distribution. In addition to the re-sampling method, several approaches have been proposed to obtain importance densities that lead to importance weights with low variance. These approaches include the auxiliary variable filter (APF) of Pitt and Shephard (1999). Furthermore, Liu and West (2001) extended the APF algorithm to consider parameter learning, which we will discuss in Section 2.2.2.

2.1.4 Auxiliary Particle Filter

Bootstrap filter uses a transition density as proposal distribution to ensure that knowledge of the current observation is incorporated into the proposal mechanism. So particles are not moved blindly into regions of the state space which are extremely unlikely in light of that observation. However it seems wasteful to resample particles at the end of time $t-1$ prior to looking at y_t. Therefore, it is natural to ask whether it is possible to employ knowledge about the next observation before resampling to ensure that particles which are likely to be compatible with that observation have a good chance of surviving, as mentioned by Whiteley and Johansen (2011).

Pitt and Shephard (1999) proposed the auxiliary variable particle filter method and try to improve some deficiencies of the SIR algorithm. The *auxiliary particle filter* (APF) algorithm invokes an auxiliary variable construction inside the sampling step to sample an auxiliary variable (a particle index) for each particle according to a distribution which weights each particle in terms of its compatibility with the upcoming observation. The idea of the APF is to mimic the availability of the optimal importance distribution by performing the re-sampling at step $t-1$ using the available measurement at time t.
In each filtering step, the algorithm consists of first drawing a sample of the particle index k which is propagated from $t-1$ into the new observation at next step t:

$$p(x_t, k|x_{t-1}, y_{1:t}) \propto p(y_t|x_t)p(x_t|x_{t-1}^k)\omega_{t-1}^k \tag{2.9}$$

Pitt (1999) approximated Equation 2.9 by the following importance density:

$$g_t(x_t, k|x_{t-1}, y_{1:t}) \propto p(y_t|\mu_t^k)p(x_t|x_{t-1}^k)\omega_{t-1}^k \tag{2.10}$$

where μ_t^k could be the mean, mode, a draw, or some other likely value of x_t given x_{t-1} such that $Pr(k = j|y_{1:t}) \propto p(y_t|\mu_t^j)\omega_{t-1}^j$.
The APF algorithm from Prado & West (2010) is as follows:

- Suppose at time $t-1$, $p(x_{t-1}|y_{1:t-1})$ is approximated by $\{(x_{t-1}, \omega_{t-1})^j; j = 1, ..., N\}$.
- For each j,
  1. Sample an auxiliary variable $k^{(j)}$ with probabilities
     $Pr(k^{(j)} = j) \propto p(y_t|\mu_t^j)\omega_{t-1}^j, j = 1, ..., N.$
     $k^{(j)}$ is the value of the j^{th} element of index vector k.
  2. Sample x_t^j from $p(x_t|x_{t-1}^{k^{(j)}})$.

3. Finally, the weights are updated to account for the mismatch between the likelihood at the actual sample and the predicted point. The new weights are calculated as: $\omega_t^j \propto p(y_t|x_t^j)/p(y_t|\mu_t^{k(j)})$.

The auxiliary variables in this algorithm are helpful in identifying the particles with larger predictive likelihoods and hence resulting in lower computation cost and becoming more efficient. A re-sampling step based on the effective sample size as introduced in the SIR algorithm above can be added.

2.2 Combined State and Parameter Estimation

In previous sections, a state space model that is function of some latent states was introduced. However, the need for more general algorithms that deal simultaneously with both state variable and unknown static parameters is pressing in many practical situations. A more general Markovian state space model that contains static parameters takes the form

$$\begin{aligned} y_t &\sim p(y_t|x_t, \theta), \\ x_t &\sim p(x_t|x_{t-1}, \theta) \end{aligned} \tag{2.11}$$

where x_t is the unobserved state vector and θ is the unobserved static or fixed parameter vector.

As pointed out by Nemeth, Fearnhead and Mihaylova (2013), while particle filter methods based on SIR and APF can work well on solving the state variables, the potential issue of particle degeneracy leaves estimation of static parameters for state space models still an open challenge. Estimation of the static parameters has received plenty of interest over the last decade. Initial approaches to this problem suggested that the parameters could be estimated by augmenting the state to

include the unknown parameters and apply a filtering approach. However, while this scheme can be employed sequentially, it quickly leads to particle degeneracy as the θ component of the augmented state comprises only of particles selected at the initialization stage (Doucet, et al. 2009).
Alternatively the problem of particle degeneracy can be partially alleviated by applying MCMC updates to θ within particle filters (see Gilks and Berzuini, 2001; Fearnhead, 2002; Storvik, 2002). The Storvik filter (2002) and the Particle Learning filter of Carvalho, et al. (2010) and Lopes, et al. (2010) are also well established filters for sequentially learning about x_t and θ.

2.2.1 Extended Particle Filters with Artificial Evolution Noise

A common trick in engineering is to include the static parameter vector θ as part of the state vector. However, the non-dynamics of the parameter quickly make the samples degenerate into one or a few different values as time t increases: The samples of θ at time t can only take the values given at the previous stage, time $t-1$. Since some of these values can become very unlikely with new observations arriving, it will quickly result in a production of distorted estimates.
Gordon, Salmond and Smith (1993) suggested adding artificial evolution noise for θ when tackling the problem of sequentially learning the static parameters of a state space model. This method can be interpreted in terms of an extended model in which the static parameters are contained and viewed as time-evolving parameters. In detail, the parameter θ is replaced by θ_t and included into an augmented state vector (x_t, θ_t). This replacement can be easily achieved by adding an independent, zero-mean normal noise to the parameter at each time point. That is,

$$\theta_{t+1} = \theta_t + \zeta_{t+1}, \qquad \zeta_{t+1} \sim N(0, \sigma_\zeta) \tag{2.12}$$

Since parameters actually do not belong to the state vector, adding noise artificially will very likely distort the validity of the approximated posterior distribution. Besides that, the variance of the noise is difficult to determine. Obviously, if the variance of noise is too large, the estimation of the posterior distributions (both states and parameter) will be inaccurate. If the variance is too small, frequently the real parameter value cannot be tracked in the model. One possible solution to this problem is make the variance σ_ζ dynamic, relatively large and decaying over time. This technique is called roughening or jittering and it is expected that this has negligible effect on the estimates.
Liu and West (2001) proposed another solution to estimate the static parameters by applying a kernel density approximation to θ, where instead of sampling parameters from a finite set of particles, parameters can now be sampled from a density. However, it is often not clear how to choose the bandwidth in the kernel density approximation, nor how this approximation impacts the accuracy of estimates of the parameters (Prado and West 2010).

2.2.2 Liu and West Particle Filter

Liu and West (2001) adapt the generic APF to sequentially re-sample and propagate particles associated with states x_t and the parameter vector θ simultaneously. More specifically, $p(x_t, \theta|y_{1:t})$ from the general state space model in (2.11) is rewritten as

$$p(x_t, \theta|y_{1:t}) \propto p(y_t|x_t, \theta)p(x_t|\theta, y_{1:t-1})p(\theta|y_{1:t-1}) \tag{2.13}$$

and $p(\theta|y_{1:t-1})$ is approximated by the mixture distribution:

$$p(\theta|y_{1:t-1}) \approx \sum_{j=1}^{N} \omega_{t-1}^j N(\theta|m_{t-1}^j, (1-a^2)V_{t-1}) \tag{2.14}$$

where a is a smoothing parameter defined as $a = (3\delta - 1)/(2\delta)$, and δ is a Discount Factor with typical value $\delta \in [0.9, 1]$. V_{t-1} is the variance of parameter samples from the last stage and the m_{t-1}^j define the locations of the mixture components. The particle set $\{(\theta_{t-1}, \omega_{t-1})^j; j = 1, ..., N\}$ is the approximation to $p(\theta|y_{1:t-1})$, $\overline{\theta}_{t-1}$ denotes the particle mean and $N(\cdot|m_{t-1}^j, (1-a)^2 V_{t-1})$ is a multivariate Gaussian density with mean m_{t-1}^j and variance/covariance matrix $(1-a)^2 V_{t-1}$.
The main attraction of Liu and West's filter is its generality as it can be implemented in any state space model. It also takes advantage of APF's re-sample-propagate framework and can be considered a benchmark in the current literature. Prado & West (2010) summarized the steps of the Liu and West algorithm as follows:
Assume at time $t-1$, $\{(x_{t-1}, \theta_{t-1}, \omega_{t-1})^j . j = 1 : N\}$ approximates $p(x_{t-1}, \theta|y_{1:t-1})$. Then for each particle j, perform the following steps:

1. Identify prior point estimate of (x_{t-1}, θ) given by (μ_t^j, m_{t-1}^j), where

$$\mu_t^j = E(x_t|x_{t-1}^j, \theta_{t-1}^j), \tag{2.15}$$

$$m_{t-1}^j = a\theta_{t-1}^j + (1-a)\overline{\theta}_{t-1}. \tag{2.16}$$

   and μ_t^j is the mean or some other likely value of x_t given x_{t-1}^j defined by the evolution equation.

2. Sample an auxiliary index $k^{(j)}$ from the set $\{1 : N\}$ with probability proportional to

$$Pr(k^{(j)} = j) \propto \omega_{t-1}^j p(y_t|\mu_t^j, m_{t-1}^j); \qquad j = 1, ..., N. \tag{2.17}$$

3. Propagate/sample a new parameter vector θ_t^j from

$$\theta_t^j \sim N(\theta|m_{t-1}^{k^{(j)}}, (1-a^2)V_{t-1}) \tag{2.18}$$

   with $V_{t-1} = \frac{1}{N-1}\sum_{j=1}^N (\theta_{t-1}^j - \overline{\theta}_{t-1})(\theta_{t-1}^j - \overline{\theta}_{t-1})'$.

4. Re-sample a value of the current state vector x_t^j from $p(x_t|x_{t-1}^{k^{(j)}}, \theta_t^j)$.

5. Compute the new weights

$$\omega_t^j \propto \frac{p(y_t|x_t^j, \theta_t^j)}{p(y_t|\mu_t^{k^{(j)}}, m_{t-1}^{k^{(j)}})}. \tag{2.19}$$

Finally $\{(x_t, \theta_t, \omega_t)^j, j = 1, ..., N\}$ approximates $p(x_t, \theta|y_{1:t})$. Note that in practice, θ may need to be transformed so that the Normal kernels in equation (2.14) are appropriate. Alternatively, other kernels, such as Beta or Gamma kernels can be used for bounded static parameters.

2.2.3 Storvik Particle Filter

To avoid the degeneracy problem, Storvik (2002) (see also Fearnhead, 2002) modifies the SIR algorithm by adding a Gibbs sampling step for θ conditional on the state trajectory. The algorithm is developed in the SIS framework and consequently inherits the theoretical justifications of SIS (Erol, et al. 2013). The approach is based on marginalizing the static parameters out of the posterior distribution such that only the state vector needs to be considered.
Suppose the posterior distribution of θ given $x_{1:t}$ and $y_{1:t}$ depends on some low-dimensional set of sufficient statistics $s_t = \mathcal{S}_t(x_{1:t}, y_{1:t})$, where $\mathcal{S}_t$ is easy to update recursively. Assume that an approximate particle set z_{t-1} is available from the posterior distribution $p(x_{1:t-1}|y_{1:t-1})$. We need an update to a new particle set z_t at time t. The approach is based on the following (Storvik 2002):

$$\begin{aligned} p(x_{1:t}, \theta|y_{1:t}) &= C * p(x_{1:t}, \theta, y_t|y_{1:t-1}) \\ &= C * p(x_{1:t-1}|y_{1:t-1})p(\theta|x_{1:t-1}, y_{1:t-1}) \\ &\quad \cdot p(x_t|x_{1:t-1}, y_{1:t-1}, \theta)p(y_t|x_{1:t}, y_{1:t-1}, \theta) \\ &= C * p(x_{1:t-1}|y_{1:t-1})p(\theta|s_{t-1})p(x_t|x_{t-1}, \theta)p(y_t|x_t, \theta) \end{aligned}$$

where $C = [p(y_t|y_{1:t-1})]^{-1}$, which is a constant not depending on $x_{1:t}$ or θ. The simplest approach would be to simulate $x_{1:t-1}$ fron $p(x_{1:t-1}|y_{1:t-1})$, θ from $p(\theta|s_{t-1})$, x_t from $p(x_t|x_{t-1}, \theta)$ and accept with probability proportional to $p(y_t|x_t, \theta)$. The key point is that the parameter θ simulated at time t does not depend on values simulated at previous time points. The Storvik algorithm has the advantage of reducing computational and memory requirements given that only the sufficient statistics need to be stored as opposed to the complete state trajectories. The algorithm is summarized as below (Storvik 2002):
For each particle $j = 1 : N$,

1. Sample the parameter particle θ^j from the proposal distributions for θ, $f_{t,1}(\theta|\theta^{(j)}_{0:t-1}; y_{1:t})$. Typically $f_{t,1}(\theta|\theta^{(j)}_{0:t-1}; y_{1:t}) = f_{t,1}(\theta|s^{(j)}_{t-1})$ in order to make computation fast.

2. Propagate and sample the state particles from the proposal distribution for x_t, $x^{(j)}_t \sim f_{t,2}(x_t|x^{(j)}_{0:t-1}, y_t, \theta)$.

3. Compute the importance weight:

$$\omega^{(j)}_t \propto \omega^{(j)}_{t-1} \frac{p(\theta|s^{(j)}_{t-1})p(x^{(j)}_t|x^{(j)}_{t-1}, \theta)p(y_t|x^{(j)}_t, \theta)}{f_{t,1}(\theta|\theta^{(j)}_{0:t-1}; y_{1:t})f_{t,2}(x^{(j)}_t|x^{(j)}_{t-1}; y_t, \theta)}. \tag{2.20}$$

4. Re-sampling:

   - Obtain the new index vector $k^{(j)}$ by resampling set $\{1 : N\}$ with weight $\{\omega^{(1)}_t : \omega^{(N)}_t\}$,
   - Put $x^j_t = x^{k^{(j)}}_t$, $s^j_{t-1} = s^{k^{(j)}}_{t-1}$ and $\omega^{(j)}_t = 1/N$.

5. Propagate the sufficient statistics $s^j_t = \mathcal{S}(s^j_{t-1}, x^j_t, y_t)$.

2.2.4 Particle Learning

Carvalho, et al. (2010) present methods for sequential filtering, particle learning (PL) and smoothing for rather general state space models. They extend Chen and Liu (2000) mixture Kalman filter (MKF) methods by allowing parameter learning and utilize a re-sample-propagate algorithm together with a particle set that includes state sufficient statistics as in the Storvik filter described in the previous section. They also show via several simulation studies that PL outperforms both the LW and Storvik filters and is comparable to MCMC samplers, even when full adaptation is considered. The advantage is even more pronounced for large values of T.

More specifically, the PL algorithm assumes that $p(\theta|x_{0:t}, y_{1:t}) = p(\theta|s_t)$, and the state sufficient statistics satisfies deterministic updating rules $s_t = \mathcal{S}(s_{t-1}, x_t, y_t)$, as in the Storvik filter from the previous subsection. The particles are represented by $\{(s_{t-1}, x_{t-1}, \theta)^{(j)}; j = 1, ..., N\}$ at time $t-1$ and the PL method updates particles at time t using the following Bayesian updating equations:

$$p(s_{t-1}, x_{t-1}, \theta|y_{1:t}) \propto p(y_t|s_{t-1}, x_{t-1}, \theta)p(s_{t-1}, x_{t-1}, \theta|y_{1:t-1})$$

and

$$p(s_t, x_t, \theta|y_{1:t}) = \int p(s_t|s_{t-1}, x_t, y_t)p(x_t|s_{t-1}, x_{t-1}, \theta, y_t) \times p(s_{t-1}, x_{t-1}, \theta|y_{1:t})ds_{t-1}dx_{t-1}$$

The PL algorithm is summarized as follows:
For each particle $j = 1 : N$,

1. Sample an auxiliary index $k^{(j)}$ with probabilities
   $Pr(k^{(j)} = j) \propto p(y_t|s_{t-1}^j, x_{t-1}^j, \theta), j = 1, ..., N.$
2. Propagate and sample the state particles $x_t^j \sim p(x_t|s_{t-1}^{k^{(j)}}, x_{t-1}^{k^{(j)}}, \theta, y_t)$.

3. Propagate sufficient statistics particles $s_t^j \sim \mathcal{S}(s_{t-1}^{k^{(j)}}, x_t^j, y_t)$.

4. Sample θ^j from $p(\theta|s_t^j)$.

2.2.5 Particle Markov Chain Monte Carlo Methods (PMCMC)

Markov Chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods are the two most popular classes of algorithms used to sample from general high-dimensional probability distributions. Unfortunately both methods have their problems especially when dealing with non-linear, non-Gaussian state-space models.

The basic idea in PMCMC is to use SMC to construct a proposal kernel for an MCMC sampler. Using MCMC steps within SMC algorithms avoids the introduction of an artificial dynamic model or of a fixed lag approximation. An approach originally proposed by Andrieu, Doucet and Holenstein (2010) consists of adding MCMC steps to re-introduce "diversity" among the particles.

The PMMH algorithm is an MCMC algorithm for state space models jointly updating the parameter θ and the states $x_{0:T}$. This algorithm runs iterations of Metropolis-Hastings or the Gibbs sampler to produce samplers which target the posterior distribution. Within each iteration, a proposed new $\theta^\star$ is first generated from a proposal $f(\theta^\star|\theta)$, and then a corresponding $x_{0:T}^\star$ is generated by running a bootstrap particle filter. However, PMCMC is extremely slow even with a small number of particles. For this reason, this algorithm is not further discussed in this thesis.

2.3 Particle Smoothing

Whereas *filtering* corresponds to estimating the distribution of the current state based upon the observations received up to the current time, *smoothing* corresponds to estimating the distribution of the state at a particular time given all of the observations up to some later time. In this sense, particle smoothers are alternatives to MCMC in state space models (Kitagawa, 1996). The trajectory estimates obtained by such methods, as a result of the additional information available, tend to be smoother than those obtained by filtering (Doucet and Johansen 2008).

The backward-simulation particle smoother (BSPS) is an smooth approach proposed by Godsill, Doucet, and West (2004) that relies on a

(i) Forward particle sampling

and

(ii) Backward particle re-sampling,

which allows to perform smoothing computation in general state space models. This method assumes the filtering has already been performed using any particle filtering, so that an approximated representation of $p(x_t|y_{1:t})$ is available at each $t = 1 : T$ via the weighted set of particles $\{(x_{0:t}, \omega_t)^j; j = 1, ..., N\}$.

To compute the marginal smoothing distribution, the joint distribution can be decomposed as

$$p(x_{0:T}|y_{1:T}) \propto p(x_T|y_{1:T}) \prod_{t=1}^{T} p(x_t|x_{t+1}, y_{1:T}). \tag{2.21}$$

By Bayes' rule and conditional independence of the state space model, we get

$$\begin{aligned} p(x_t|x_{t+1}, y_{1:T}) &= p(x_t|x_{t+1}, y_{1:t}) \\ &= \frac{p(x_t|y_{1:t})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} \\ &\propto p(x_t|y_{1:t})p(x_{t+1}|x_t). \end{aligned}$$

Then it is possible to obtained the modified particle approximation

$$p(x_t|x_{t+1}, y_{1:T}) \approx \sum_{j=1}^{N} \overleftarrow{\omega_{t|t+1}}^j \delta_{x_t^j}(x_t) \tag{2.22}$$

with

$$\overleftarrow{\omega_{t|t+1}}^j = \frac{\omega_t^j p(x_{t+1}|x_t^j)}{\sum_{i=1}^{N} \omega_t^i p(x_{t+1}|x_t^i)}$$

So the weights for the smoothing approximation are modified by $p(x_{t+1}|x_t)$. Leading to the following particle smoothing algorithm as in Prado & West (2010):

1. After implementing all the filtering steps, at time T, approximate $p(x_T|y_{1:T})$ by choosing $\{(\overleftarrow{x_T}, \overleftarrow{\omega_T})^j; j = 1, ..., N\} = \{(x_T, \omega_T)^j; j = 1, ..., N\}$

2. For $t = (T-1) : 0$,

   - Calculate $\overleftarrow{\omega_{t|t+1}}^j \propto \omega_t^j p(\overleftarrow{x_{t+1}}|x_t^j)$ for $j = 1, ..., N$.
   - Re-sample by choosing $\overleftarrow{x_t}^j = x_t^j$ with probability $\overleftarrow{\omega_{t|t+1}}^j$.

3. Then $\{\overleftarrow{x_t}^j,\ j = 1, ..., N\}$ is an approximate realization from the smoothing distribution $p(x_t|y_{1:T})$

Steps 1 and 2 can be repeated several times to obtain independent approximate realizations of $p(x_t|y_{1:T})$.

Chapter 3

Simulation Study for the First Order Dynamic Linear Model

In this chapter we illustrate the algorithms introduced in Chapter 2 using simulated data from a first order dynamic linear model with unknown observation variance and to evaluate the performance of the particle filters when one static parameter is unknown. This DLM has the same format as the model discussed in Section 1.4.4 but with constant observation and evolution variance.

$$Y_t = x_t + v_t, \qquad v_t \sim N(0, V), \tag{3.1}$$

$$x_t = x_{t-1} + \omega_t, \qquad \omega_t \sim N(0, W). \tag{3.2}$$

To simplify the computation, I assume the state variance W is provided and is relatively small compared to the observation variance V. We have chosen this model so that we can easily apply different methodologies presented in this thesis and compare their performance.

Estimated posterior distributions from particle filters are compared with the posterior distributions obtained from a full MCMC run at each time point using a

large number of iterations, M=100,000. The number of particles has been set to N=5000, such that a reasonable agreement with the MCMC runs could be reached.

3.1 MCMC Algorithm

While sampling the parameters depends heavily on the model and priors being used, the state vector sequence can be generated from its full conditional distribution using the so-called forward filtering backward sampling (FFBS) algorithm (Carter and Kohn 1994). This algorithm is basically a simulation version of the Kalman smoother, consisting in running the Kalman Filter first, followed by a backward recursion to generate all the states from the final time T to time 0. FFBS is well suited to deal with state space models that are conditionally Gaussian. As illustrated by West and Harrison (1997), a fully conjugate Bayesian analysis is capable of providing specific equations for the posterior distribution of the observation variance V.

3.1.1 Prior for the Observation Variance

In many univariate structural models (local level, local linear trend, seasonal factor components), the only unknown parameters are the observation and system variances. In this case, I assume independent Inverse-Gamma priors for these variances since it provides an easy way to set up a Gibbs sampler - conditionally conjugate model. Besides, it is convenient to work with the precision $\phi = 1/V$ rather than with the variance V.

A conjugate prior for ϕ is such that it has a Gamma density with parameters (a_0, b_0) and denoted by $\phi \sim Gamma(a_0, b_0)$. Here b_0 is the inverse scale, or so

called rate parameter so the mean of this Gamma distribution is a_0/b_0. Given the observations $(y_1, y_2, ...y_T)$, we can update our opinion about ϕ through the computation of the posterior density. Using Bayes formula, we can compute the posterior density for ϕ given $x_{1:T}$ and $y_{1:T}$. For the model of equations 3.1 and 3.2:

$$\begin{aligned}
\pi(\phi|x_{1:T}, y_{1:T}) &\propto p(\phi) \cdot \prod_{t=1}^{T} p(y_t|x_t, \phi) \\
&\propto (\phi)^{a_0-1} exp(-\phi b_0) \cdot (\phi)^{T/2} exp\left\{-\frac{1}{2}\sum_{t=1}^{T}\phi(y_t - x_t)^2\right\} \\
&\propto (\phi)^{a_0+T/2-1} exp\left\{-\phi\left(b_0 + \frac{1}{2}\sum_{t=1}^{T}(y_t - x_t)^2\right)\right\}
\end{aligned}$$

which shows the conjugate posterior density of

$$p(\phi|x_{1:T}, y_{1:T}) = Gamma\left(a_0 + T/2, b_0 + \frac{1}{2}\sum_{t=1}^{T}(y_t - x_t)^2\right).$$

One of the most commonly used non-informative prior for the variance V is the Inverse-Gamma with parameters $a_0 = 0.001$ and $b_0 = 0.001$ (Gelman 2006). For the states $x_{1:T}$, we need a prior for x_0:

$$x_0 \sim N(m_0, C_0)$$

with $m_0 = 0$ and $C_0 = 100$ or a larger value of C_0 to make the prior non-informative.

3.1.2 Gibbs Sampling

Since we can easily get the full conditional densities for this first order DLM, a Gibbs sampler is good enough to run the MCMC and get the approximation of the joint posterior density $\pi(x_{1:T}, V|y_{1:T})$. The Gibbs sampling iterations are summarized as follows:

1. Set initial value for the variance, $V^{(0)}$. This initial value could be generated from the prior density. Alternatively, one could use the sample variance of all the observations: $V^{(0)} = var(y_{1:T})$. The initial value of x_0 can be a random draw from its prior distribution.

2. For iteration $m = 1, 2, ...$

   (a) Apply the FFBS algorithm as discussed in Section 1.4.4 to generate samples of the state parameter $x_{1:T}^{(m)}|y_{1:T}$ given $V^{(m-1)}$.

   (b) Generate $V^{(m)}$ from $IG\left(a_0 + T/2, b_0 + \frac{1}{2}\sum_{t=1}^{T}(y_t - x_t^{(m)})^2\right)$.

After convergence is reached, the resulting samples are draws from the posterior density. Theoretically, values from the posterior density are only obtained after running the algorithm an infinite number of iterations. In practice, a visual inspection of trace plots can be used to check if the chains empirically converged to a stable distribution within the parameter space. We may also look at density plots and check the autocorrelation between draws of the Markov chain. We would expect the k^{th} lag autocorrelation to be smaller as the value k increases. Furthermore, it is a standard practice to discard the initial iterations or burn-in as they could be strongly influenced by the starting values and may not provide good information about the target distribution.

3.2 Particle Filter Algorithm

As discussed in section 2.2.1, for the BS and APF particle filters, we will include the static parameters in the state vector and make it evolve with an artificial variance that decays over time. A log transformation on V is considered in order to use a Normal distribution proposal. The details to approximate the posterior

density of the parameters with particle filters for a first order DLM are provided next.

3.2.1 BS Filtering

1. Set initial values:

   (a) Pick N samples of $log(V)$ from a $N\left(log[var(y_{1:5})], \sigma_0\right)$. Here $var(y_{1:5})$ is the sample variance of the first 5 observations, other number of initial observations can be chosen as well (e.g. 10 or 20). Different σ_0 values are considered and an optimal value was picked in order to reach convergence fast.

   (b) Pick N samples of $x_t^{(0)}$ based on the prior density, $x_t^{(0)} \sim N(m_0, C_0)$.

2. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   i. Sample new particles from the dynamic model:
   First take jittering for the dynamic variance: $\sigma_t = 0.9\sigma_{t-1}$ and then propagate it:

   $$log(V_t^j) \sim N(log(V_{t-1}^j), \sigma_t) \tag{3.3}$$

   $$x_t^j \sim N(x_{t-1}^j, W). \tag{3.4}$$

   ii. Calculate the weights:

   $$\omega_t^j \propto p(y_t|x_t^j, V_t^j), j = 1, ..., N \tag{3.5}$$

   given that $p(y_t|x_t^j, V_t^j) = N(y_t|x_t^j, V_t^j)$.

   iii. Re-sample x_t^j and V_t^j based on the new weights.

3.2.2 APF Filtering

1. Set initial values as in the BS filtering case.

2. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   First use jittering to define a dynamic variance: $\sigma_t = 0.9\sigma_{t-1}$.

   i. Sample an auxiliary variable by first computing the weight:

   $$\mathbf{w}_t^j \propto p(y_t|\mu_t, V_{t-1}^j). \tag{3.6}$$

   We take μ_t to be the mean of the distribution of x_t given x_{t-1}^j, which is simply $\mu_t = x_{t-1}^j$ due to the Normality of $p(x_t|x_{t-1})$. Then we obtain the N dimensional auxiliary index vector k based on sampling over the set $\{1 : N\}$ using the N weights.

   ii. Propagate the new particles from the original particles under the auxiliary index vector k as:

   $$\begin{aligned} log(V_t^j) &\sim N\Big(log(V_{t-1}^{k^{(j)}}), \sigma_t\Big) \\ x_t^j &\sim N\Big(x_{t-1}^{k^{(j)}}, W\Big), \end{aligned}$$

   where $k^{(j)}$ represent the index value from the j^{th} element of the auxiliary index vector k.

   iii. Compute a new weight using:

   $$\omega_t^j \propto p(y_t|x_t^j, V_t^j)/\mathbf{w}_t^{k^{(j)}},$$

   where $p(y_t|x_t^j, V_t^j) = N(y_t|x_t^j, V_t^j)$.

   iv. Re-sample x_t^j, V_t^j based on the new weights.

3.2.3 LW Filtering

1. Set initial values as in the BS filtering case.

2. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   i. Calculate the sample variance from the particles of V at $t-1$:

$$\begin{aligned} \mu_{log(v)} &= \frac{1}{N}\sum_{i=1}^{N} log(V_{t-1}^{i}), \\ vpar_{t-1} &= \frac{1}{N-1}\sum_{i=1}^{N}\left[log(V_{t-1}^{i}) - \mu_{log(v)}\right]^2. \end{aligned} \tag{3.7}$$

   Calculate the prior mean estimate based on the particles at $t-1$:

$$\mu_t^j = E(x_t|x_{t-1}^j) = x_{t-1}^j, \tag{3.8}$$

$$m_{t-1}^j = a log(V_{t-1}^j) + (1-a)\mu_{log(v)}. \tag{3.9}$$

   ii. Compute the first stage weights:

$$\begin{aligned} \mathbf{w}_t^j &\propto p(y_t|\mu_t^j, m_{t-1}^j) \\ &= N(y_t|\mu_t^j, m_{t-1}^j) = N(y_t|x_{t-1}^j, m_{t-1}^j). \end{aligned}$$

   And obtain the index vector k based on sampling the set $\{1:N\}$ using these weights.

   iii. Perform a parameter re-sample of V_t^j:

$$log(V_t^j) \sim N\left(m_{t-1}^{k^{(j)}}, (1-a^2)vpar_{t-1}\right). \tag{3.10}$$

   Perform the propagation of the state x_t^j

$$x_t^j \sim N\left(x_{t-1}^{k^{(j)}}, W\right). \tag{3.11}$$

   iv. Compute the second stage weights

$$\omega_t^j \propto p(y_t|x_t^j, V_t^j)/\mathbf{w}_t^{k^{(j)}}, \tag{3.12}$$

   where $p(y_t|x_t^j, V_t^j) = N(y_t|x_t^j, V_t^j)$.

   v. Do a re-sample of x_t^j and V^j based on the second stage weights.

3.2.4 PL Filtering

1. Set initial values as in the BS filtering case.

2. The sufficient statistics for this model are:
$$s_t = \left[\frac{t}{2}, \frac{1}{2}\sum_{l=1}^{t}(y_l - x_l)^2\right].$$

3. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   i. Obtain an auxiliary index vector by computing the weights:
   $\mathbf{w}_t^j \propto p(y_t|x_{t-1}^j, V_{t-1}^j, W)$,
   with $p(y_t|x_{t-1}^j, V_{t-1}^j, W) = N(y_t|x_{t-1}^j, V_{t-1}^j + W)$.
   And get the index vector k by sampling over the set $\{1 : N\}$ with these weights.

   ii. Propagate the state particles x_t^j by
   $$\begin{aligned} x_t^j &\sim p(x_t|s_{t-1}^j, x_{t-1}^j, V^j, W, y_t) \\ &= p(x_t|x_{t-1}^j, V^j, W, y_t) \\ &\propto p(y_t|x_t, V^j, W)p(x_t|x_{t-1}^j, V^j, W) \\ &\propto N\left\{\frac{y_t W + x_{t-1}V^j}{V^j + W}, \frac{V^j \cdot W}{V^j + W}\right\}. \end{aligned}$$
   See Appendix A.4 for the proof of this step.

   iii. Update the sufficient statistics
   $$\begin{aligned} s_t &= f(s_{t-1}, x_t, y_t) \\ &= f\left(\frac{t-1}{2}, \quad \frac{1}{2}\sum_{l=1}^{t-1}(y_l - x_l)^2, x_t, y_t\right) \\ &= \left[\frac{t-1}{2} + \frac{1}{2}, \quad \frac{1}{2}\sum_{l=1}^{t-1}(y_l - x_l)^2 + (y_t - x_t)^2\right] \\ &= \left[\frac{t}{2}, \quad \frac{1}{2}\sum_{l=1}^{t}(y_l - x_l)^2\right]. \end{aligned}$$

iv. Sample the static parameter from $p(\theta|s_t^j)$ as:

$$V|s_t \sim IG\left(a_0 + t/2, \quad b_0 + \frac{1}{2}\sum_{l=1}^{t}(y_l - x_l)^2\right)$$

3.3 Simulation Study

In this section, a simulation case to approximate the posterior densities of the state parameters x_t and observation variance V is studied. A dataset is generated from the first order DLM model described in Equations 3.1 and 3.2 with fixed parameters $V = 1$, $W = 0.01$ and $x_t = x_{t-1} + \omega_t$ with $x_0 = 25$ and $T = 200$. We compare the approximation from the MCMC and all particle filter methods by sequentially updating the filtering density $p(\mu_{1:t}|y_{1:t}), t = 11, ..., T$ as new observations are processed. The detail procedure is as follows: use $t = 1 : 10$ as a warm-up period and run the MCMC and particle filters based on these first 10 observations. For a subsequent period ($t = 11, 12, ...T$), run the MCMC on $y_{1:t}$ and draw posterior samples. This gives an approximation to the filtering distribution $p(x_t|y_{1:t})$. We start the comparison from $t = 11$ since a smaller number of observations will not provide enough information to approximate the posterior density. For particle filters, we retain particles from the previous stage running on $y_{1:t}$ and update these particles with a new observation y_{t+1}. The updated particles are a posterior density approximation for $p(x_{t+1}|y_{1:t+1})$. For MCMC simulation, the total number of iterations is set to $M = 4,000$ and 3000 samples for burn-in without thinning. $N = 5000$ particles were used for all particle filter methods. The results are shown in Figures 3.2 to 3.3.

Figure 3.2 shows the approximation of the posterior density of the state parameter $p(x_t|y_{1:t}), t = 11, ..., T$ from different methods. In plot (a), the dots represent the observations generated from the first order DLM model. The solid line in red color

are the true values of the state x_t. The filled gray area represents the 95% credible interval of the MCMC simulation and the solid black line is the posterior median value at each time point. The plot shows that the median values from the MCMC method follow a similar behavior as the true values of the state parameter x_t. The 95% credible interval of the MCMC method covers the true state parameter for most of the time points except at $t = 80$ to $t = 95$. Plot (b) illustrates the comparison between the MCMC and the BS particle filter approximation. Just like in plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line represents the posterior median values. The solid line in blue are the posterior median values of the BS particle filter approximation and the dashed lines in blue give the 95% credible interval. The 95% credible interval from the BS particle filter matches with the one from MCMC and the BS particle filter achieves almost the same result as the MCMC method. Plot (c) shows a comparison between the MCMC and the APF particle filter approximation. Plot (d) illustrates the comparison between the MCMC and the LW particle filter approximation. Plot (e) illustrates the comparison between the MCMC and the PL particle filter approximation. In plots (c) and (e), the solid black line representing the posterior median values from the MCMC method is not easily identified since it is overlapping with the solid blue line representing the posterior median values from the particle filter method. Plots (c), (d) and (e) prove that the APF, LW and PL particle filters achieve similar results as the MCMC method in estimating the state parameter x_t. Figure **??** shows estimates of the posterior density of the observational variance parameter V, $p(V|y_{1:t})fort = 11, ..., T$. In plot (a), the filled gray area represents the 95% credible interval of the MCMC simulation and the solid black line are the posterior median values. The solid line in blue stands for the posterior median values of the BS particle filter approximation and the dashed lines in blue cover the 95% credible interval. The MCMC estimates match with the BS particle filter estimates through all time

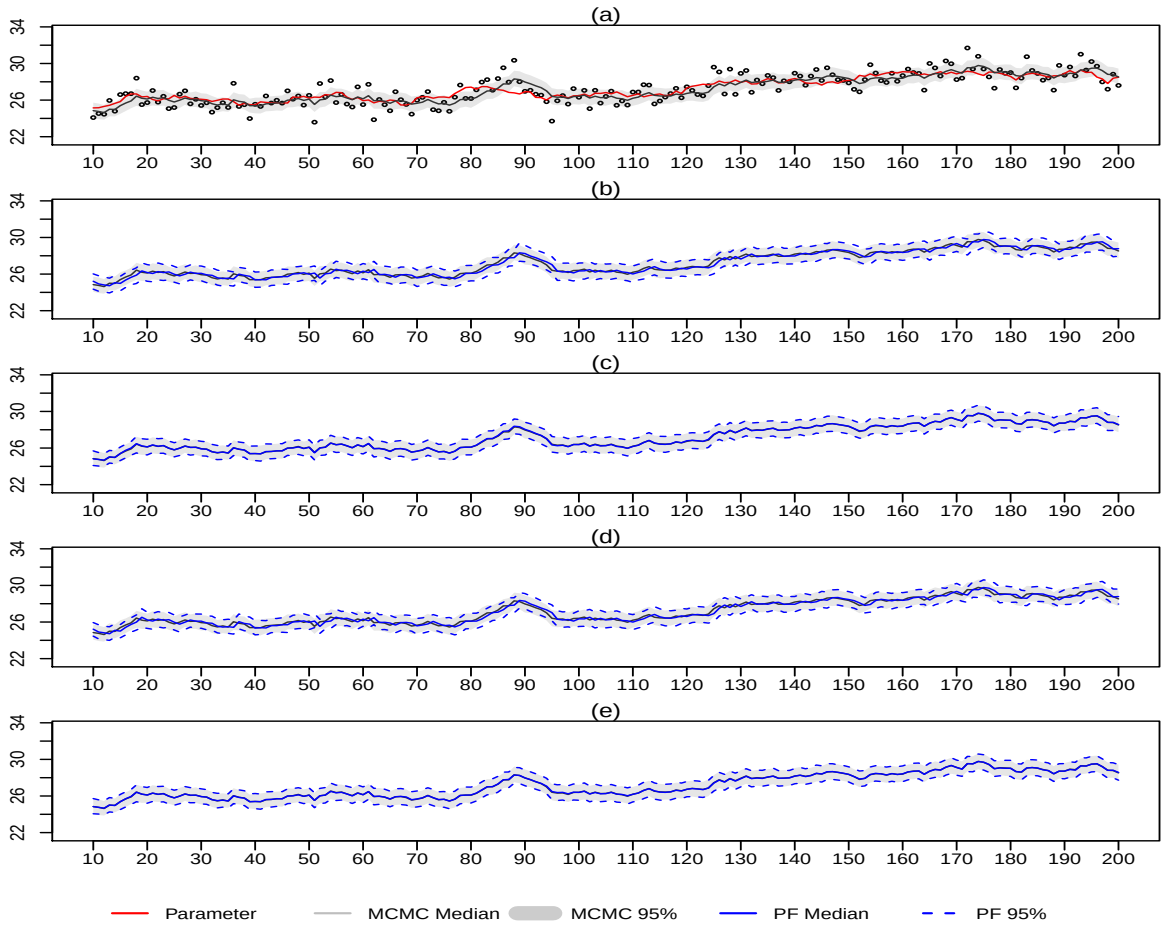

Figure 3.1: Estimation of the state parameter x_t of a first order DLM. (a) True parameter vs MCMC, (b) MCMC vs BS, (c) MCMC vs APF, (d) MCMC vs LW, (e) MCMC vs PL.

points, $t = 11, ..., 200$. The posterior median values and the 95% credible interval from both the MCMC and BS particle filter methods tend to agree with each other. Plot (b) illustrates the comparison between the MCMC and the APF particle filter approximation. This plot clearly shows the APF particle filter achieves a similar result as the BS particle filter but tends to provide a more stable 95% credible interval. Plot (c) illustrates the comparison between the MCMC and the LW particle filter approximation. Plot (d) illustrates the comparison between the

MCMC and the PL particle filter approximation. Among all these methods, the PL particle filter method provides the best match with , while the LW method provides the second best match to MCMC. Other particle filter methods also provide comparable estimates of the posterior distribution but the LW particle filter generates the smallest 95% credible interval. Figure 3.3 shows histograms

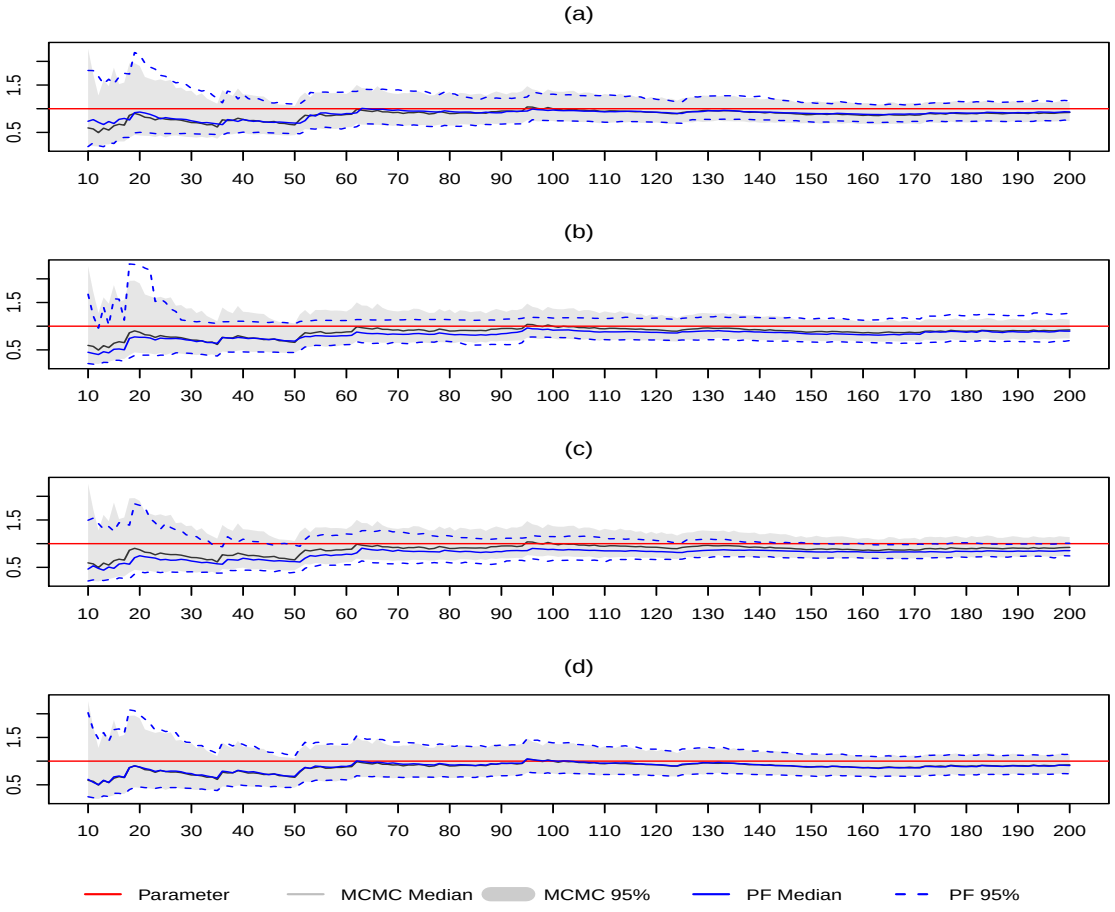

Figure 3.2: Estimation of the state parameter x_t of a first order DLM. (a) True parameter vs MCMC, (b) MCMC vs BS, (c) MCMC vs APF, (d) MCMC vs LW, (e) MCMC vs PL.

of the samples of the posterior density approximation at the last time point T, $p(x_T|y_{1:T})$ and $p(V|y_{1:T})$ for all methods considered respectively. Histograms have

the same range in order to provide a clear comparison among different methods. Histograms corresponding to $p(x_T|y_{1:T})$ on the left panel show all methods provide similar samples of the state parameter conditional on all the data. On the right panel, all the histograms of the samples of the posterior density approximation for $p(V|y_{1:T})$ contain the true parameter value of 1. The result from the PL method is graphically closer to the MCMC result while the LW method provides the smallest credible.

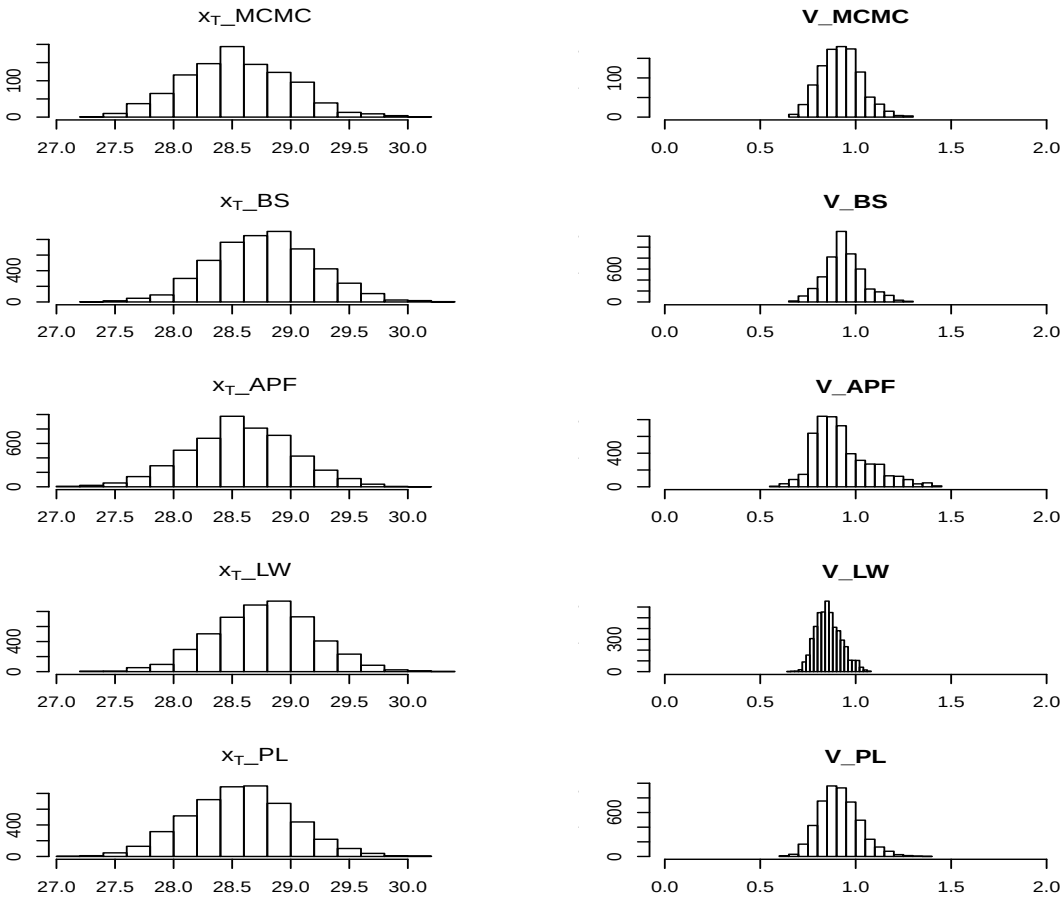

Figure 3.3: Histograms of posterior samples for a first order DLM.

Chapter 4

The Generalized Extreme Value (GEV) Distribution

Extreme value theory has been applied to assess the extreme behavior of random variables in various fields from natural sciences (e.g., extreme temperature) to financial econometric (e.g., risk management). In recent years, dynamic extreme value models have attracted some attention to investigate time-dependence change of extremes. A MCMC algorithm is generally used to estimate the posterior distributions of the parameters of this model in a Bayesian framework.

In this chapter, we apply the MCMC and particle filter approaches mentioned in this thesis to dynamic GEV models. Since MCMC is a well recognized approach to identify the dynamic state and static parameters for GEV distribution, I consider the MCMC posterior approximation as a benchmark to test performance on model estimation of particle filter approximations.

4.1 GEV Distribution

As mentioned by Adlouni and Ouarda (2009), extreme value analysis allows the interpretation of past records and the inference about future probabilities of occurrence of extreme events, such as floods, extreme rainfalls, or high wind gusts. Extreme values are often represented by the maximum value of the variable of interest over a given time period, such as a day, a month or even a year.
The generalized extreme value (GEV) distribution, also known as the *von Mises type extreme value distribution* or the *von Mises-Jenkinson type distribution* (Jenkinson, 1955), is a family of continuous probability distributions developed within the extreme value theory that combines the Gumbel, Frèchet and Weibull families also known as type I, II and III extreme value distributions (Coles 2001).
Let y_t, for $t = 1, \ldots, T$, be the maximum of a block of observations over a unit of time (e.g., a year, a month, a day). Then y_t follows a GEV distribution, denoted here by $y_t \sim GEV(\mu, \sigma, \xi)$. Its cumulative distribution function (cdf) H is given by

$$H(y) = Pr(y_t \leq y) = \exp\left\{-\left(1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right)^{-\frac{1}{\xi}}\right\}, \tag{4.1}$$

where $\mu \in \mathbb{R}$, $\sigma > 0$ and $\xi \in \mathbb{R}$ are the location, scale and shape parameters, respectively, and $1 + \xi(y_t - \mu)/\sigma > 0$. More specifically, y_t varies in $[\mu - \sigma/\xi, \infty)$, $(-\infty, \mu - \sigma/\xi]$ and $\mathbb{R}$ for $\xi > 0$ (Frèchet), $\xi < 0$ (Weibull) and $\xi = 0$ (Gumbel), respectively. The case of $\xi > 0$ corresponds to a heavy-tail, $\xi < 0$ is the case of a finite upper endpoint and $\xi = 0$ is the case of an exponentially decreasing tail (Kotz and Nadarajah, 2000). $H(y)$ arises through a limiting argument for block maxima as the block size goes to infinity in the so-called *Extreme Type Theorem* that is discussed for example, in Coles (2001).

The density function is, consequently,

$$f(y|\mu,\sigma,\xi) = \frac{1}{\sigma}\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{(-1/\xi)-1} \cdot exp\left\{-\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{(-1/\xi)}\right\}$$

for $y > \mu - \sigma/\xi$ in the case $\xi > 0$, and for $y < \mu + \sigma/\xi$ in the case $\xi < 0$. The density is 0 outside these ranges.

In the case $\xi = 0$ the density is positive on the whole real line and equal to

$$f(y|\mu,\sigma,\xi) = \frac{1}{\sigma}\exp\left[-\left(\frac{y-\mu}{\sigma}\right)\right]\exp\left\{-\exp\left[\left(-\frac{y-\mu}{\sigma}\right)\right]\right\}.$$

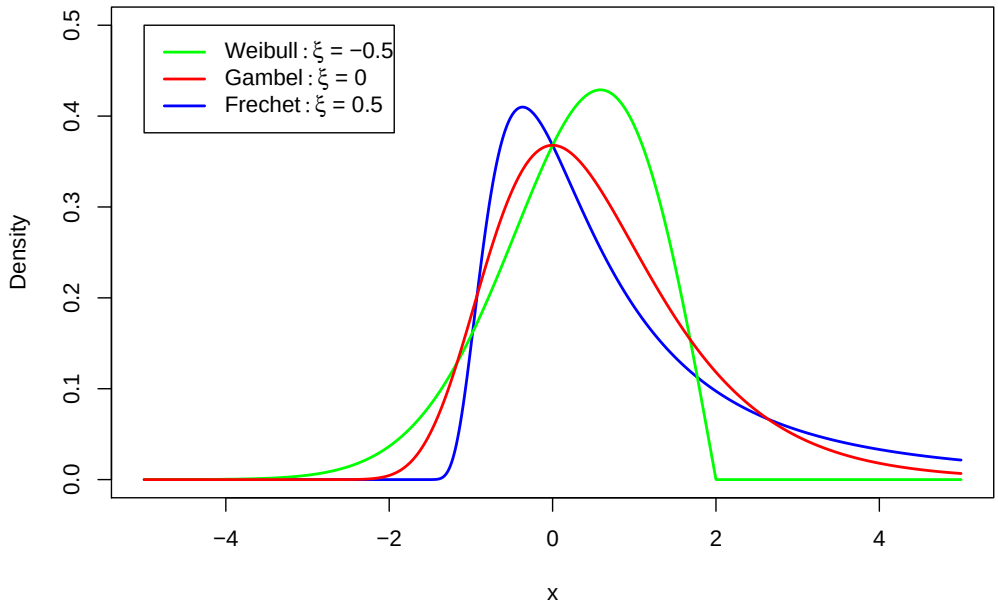

Figure 4.1: Probability density curves of the GEV distribution for 3 values of the shape parameter with $\mu = 0$ and $\sigma = 1$.

Figure 4.1 shows the pdfs of the GEV distribution for $\xi = -0.5, 0, 0.5$ respectively with $\mu = 0$ and $\sigma = 1$. We can clearly see the difference between the three types

of GEV distributions. However, given a shape value of 0.1 and -0.1, the difference between the three types of GEV distributions is less evident as shown in Figure 4.2.

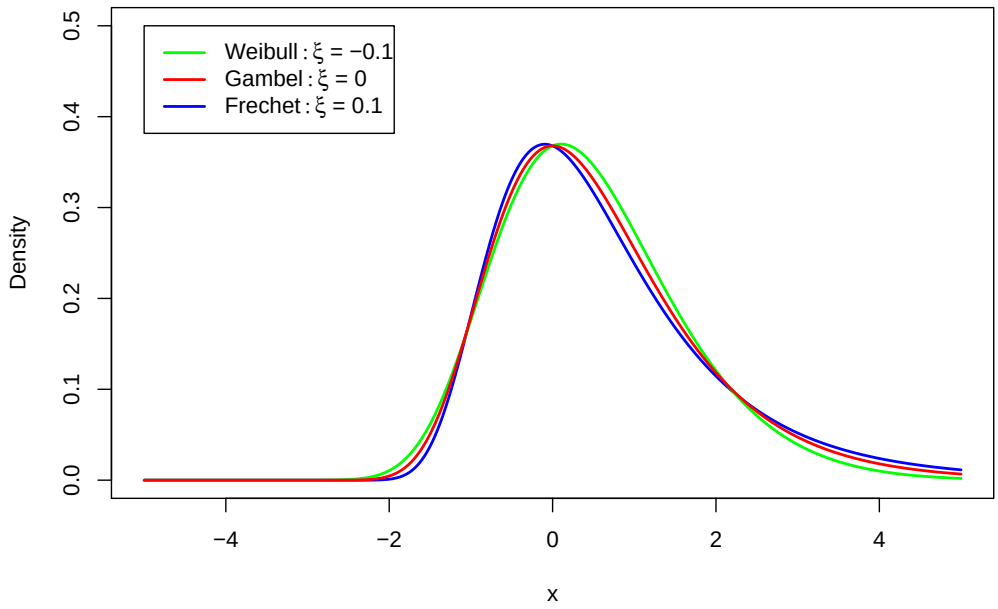

Figure 4.2: Probability density curves of the GEV distribution for 3 values of the shape parameter with $\mu = 0$ and $\sigma = 1$.

4.2 MCMC for the GEV Distribution with Time Components

In order to include time varying components, the standard extreme value model can be enhanced by assuming a functional form, usually a time dependence model, for each of its parameters. Gaetan and Grigoletto (2004) assume that the parameters

$\gamma_t = (\mu_t, \log \sigma_t, \xi_t)$ are time-varying and related to a d-dimensional state vector X_t as

$$\gamma_t = Z_t X_t, \tag{4.2}$$

where Z_t is a known $3xd$ matrix. The state varies according to a standard linear evolution with Gaussian errors

$$X_t = GX_{t-1} + w_t, \qquad w_t \sim N(0, W) \tag{4.3}$$

where G is a known d by d transition matrix and the initial state is modeled as $X_0 \sim N(m_0, C_0)$.

Gaetan and Grigoletto (2004) also consider that the behavior of the location parameter μ_t can be decomposed in a long-term trend, and/or a seasonal component and/or a term which includes covariate dependence. In their case study about the Women's 3000m athletic record, a dataset that I consider in a later chapter, they assume fixed scale and shape parameters while consider the time behavior of μ_t is described by a first-order random walk (RW1),

$$\mu_t = \mu_{t-1} + \omega_t \tag{4.4}$$

or a second-order random walk (RW2)

$$\mu_t = 2\mu_{t-1} - \mu_{t-2} + \omega_t \tag{4.5}$$

where the errors ω_t are independent and follow a $N(0, V)$ distribution.

My study will be based on the model suggested by Gaetan and Grigoletto (2004) with a RW1 or RW2 time evolution on the state μ_t. We need to estimate the posterior distribution $p(\mu_{1:t}, \sigma, \xi | y_{1:T})$. In the next section, we develop a MCMC algorithm to estimate the parameters of the time-dependent GEV model where the state variables μ_t follow the RW1 process. To use a Gaussian distribution, a log transformation is employed for the scale parameter σ.

A problem for the MCMC is how to set the initial condition that direct to fast convergence. A good initial condition that is close to the true parameter values can guarantee rapid convergence of the MCMC and particle filter approaches. However, when the initial conditions deviate far from the true values, the probability of the particles approaching the true states can be very low, especially in the early time stages. Although theoretically convergence occurs regardless of the starting point, a bad starting point may make convergence difficult or even impossible to achieve. A possible approach to set the the starting values of μ_0 is through trial and error, or with the mean value of a few observations, such as

$$\mu_t^{(0)} = \frac{1}{10}\sum_{t=1}^{10}(y_t).$$

The initial values of the scale and shape parameters are generated using a maximum likelihood estimation of the GEV distribution by considering that location parameter μ_t, constant across time.

4.2.1 Prior Distributions

Coles and Powell (1996) argue that eliciting priors in terms of the GEV parameters is not necessarily the most sensible approach to express prior beliefs on this distribution. However, without any previous information and given the computational complexities of a time-varying GEV model, we need to choose priors on parameters rather than on quantiles or differences of quantiles as suggested by these authors. An important but reasonable assumption is that the parameters are independent, allowing us to define their priors separately.
We adopt independent Normal density priors for all states and parameters, so the joint prior distribution for $p(\mu_{1:T}, \sigma, \xi, V)$ is given by:

- $\mu_0 \sim N(M_\mu, V)$,

- $log(\sigma) \sim N(M_\sigma, V_\sigma)$,
- $\xi \sim N(M_\xi, V_\xi)$, and
- $log(V) \sim N(M_V, V_V)$.

where $M_\mu, M_\sigma, M_\xi, M_V$ are the prior means and V_σ, V_ξ, V_V are the prior variances respectively. We only define the prior for μ_0 since for the other location parameters $\mu_{1:T}$ the prior can be obtained through the time behavior defined in Equation 4.4 or 4.5. For the evolution variance V an Inverse-Gamma prior can be used as in the previous chapter.

4.2.2 Posterior Distributions

Let $\{y_1, y_2, ..., y_T\}$ be independent realizations from a GEV distribution conditional to all model parameters. The likelihood function is:

$$
\begin{aligned}
&L(\mu_{1:T}, \xi, , \sigma, \mid y_{1:T}) = \prod_{t=1}^{T} p(y_t|\mu_t, \xi, \sigma) \\
&= \prod_{t=1}^{T} \frac{1}{\sigma}\left[1+\xi\left(\frac{y_t-\mu_t}{\sigma}\right)\right]_+^{-(1+\frac{1}{\xi})} \exp\left\{-\left[1+\xi\left(\frac{y_t-\mu_t}{\sigma}\right)\right]_+^{-\frac{1}{\xi}}\right\}.
\end{aligned}
$$

Using Bayes theorem, the posterior distribution is proportional to the product of the likelihood function and the prior distribution.

$$
\begin{aligned}
&p(\mu_{1:T}, \xi, \sigma, V|y_{1:T}) \propto p(y_{1:T}|\mu_{1:T}, \xi, \sigma, V) \times p(\mu_{1:T}, \xi, \sigma, V) \\
&\propto p(y_{1:T}|\mu_{1:T}, \xi, \sigma, V) \times p(\mu_{1:T}|\xi, \sigma, V) \times p(\xi, \sigma, V) \\
&\propto \left\{\prod_{t=1}^{T} p(y_t|\mu_t, \xi, \sigma)\right\} \times p(\mu_0) \times \left\{\prod_{t=1}^{T} p(\mu_t|\mu_{t-1}, V)\right\} \times p(\xi) \times p(\sigma) \times p(V) \\
&\propto \left\{\prod_{t=1}^{T} p(y_t|\mu_t, \xi, \sigma)\right\} \times \exp\left\{-\frac{(\mu_0 - M_\mu)^2}{2\,V_\mu}\right\} \times \left\{\prod_{t=1}^{T} \exp\left\{-\frac{(\mu_t - \mu_{t-1})^2}{2\,V}\right\}\right\}
\end{aligned}
$$

$$\times \exp\left\{-\frac{(log(V)-M_V)^2}{2\,V_V}\right\} \times \exp\left\{-\frac{(\xi-M_\xi)^2}{2\,V_\xi}\right\} \times \exp\left\{-\frac{(\log\sigma-M_\sigma)^2}{2\,V_\sigma}\right\}$$

The full conditional distributions for each parameter ξ, σ and V:

$$p(\xi|y_{1:T},\mu_{1:T},\sigma,V) \propto \left\{\prod_{t=1}^{T} p(y_t|\mu_t,\xi,\sigma)\right\} \times p(\xi),$$

$$p(\sigma|y_{1:T},\mu_{1:T},\xi,V) \propto \left\{\prod_{t=1}^{T} p(y_t|\mu_t,\xi,\sigma)\right\} \times p(\sigma),$$

$$p(V|y_{1:T},\mu_{1:T},\xi,\sigma) \propto \left\{\prod_{t=1}^{T} p(\mu_t|\mu_{t-1},V)\right\} \times p(V).$$

The full conditional distributions for each $\mu_t, t = 2, ...(T-1)$ is given by:

$$p(\mu_t|y_{1:T},\{\mu_{-t}\},\xi,\sigma,V) \propto p(y_t|\mu_t,\xi,\sigma) \times p(\mu_t|\mu_{t-1},V) \times p(\mu_{t+1}|\mu_t,V)$$

where $\{\mu_{-t}\}$ is the set of location parameters without μ_t, that is

$$\{\mu_{-t}\} = \{\mu_1, ..., \mu_{t-1}, \mu_{t+1}, ..., \mu_T\}.$$

The full conditional distributions for μ_1 is:

$$p(\mu_1|y_{1:T},\{\mu_{2:T}\},\xi,\sigma,V) \propto p(y_1|\mu_1,\xi,\sigma) \times p(\mu_2|\mu_1,V).$$

The full conditional distributions for μ_T is given by:

$$p(\mu_T|y_{1:T},\{\mu_{1:T-1}\},\xi,\sigma,V) \propto p(y_T|\mu_T,\xi,\sigma) \times p(\mu_T|\mu_{T-1},V).$$

Posterior draws for μ_t, V, σ and ξ can be obtained through full conditional draws of each parameter based on the Metropolis-Hastings (M-H) algorithm. For example, to draw the shape parameter ξ with a M-H step, at iteration $m+1$:

1. Sample ξ^{m+1} from a Normal distribution centered at ξ^m which defines a symmetric proposal distribution based on a Random Walk so our M-H step reduces to a pure Metropolis step.

2. Compute $\alpha(\xi^m, \xi^{m+1}) = min\left\{1, \frac{p(\xi^{m+1})}{p(\xi^m)}\right\}$, where $p(\xi^{m+1})$ is the full conditional posterior distribution evaluated at the generated value ξ^{m+1}, and $p(\xi^m)$ is full conditional posterior evaluated at the previous sampled ξ^m.

3. Generate $u \sim U(0,1)$. If $u < \alpha(\xi^m, \xi^{m+1})$, we accept the proposed value ξ^{m+1} as our current point in the chain. Otherwise, we reject ξ^{m+1} and keep the previous point ξ^m.

The Estimation of theμ_t, V and σ follow a similar procedure. A difficulty here is how to find an appropriate symmetric proposal distribution for μ_t:
The full conditional distributions for $\mu_t, t = 2, ...(T-1)$ can be simplified as:

$$
\begin{aligned}
& p(\mu_t|\mu_{t-1}, V) \times p(\mu_{t+1}|\mu_t, V) \\
& \propto \exp\left\{-\frac{(\mu_t - \mu_{t-1})^2}{2\,V}\right\} \times \exp\left\{-\frac{(\mu_{t+1} - \mu_t)^2}{2\,V}\right\} \\
& \propto \exp\left\{-\frac{2\mu_t^2 - 2\mu_t(\mu_{t-1} + \mu_{t+1})) + \mu_{t-1}^2 + \mu_{t+1}^2}{2\,V}\right\} \\
& \propto \exp\left\{-\frac{\mu_t^2 - 2(\mu_t)[(\mu_{t-1} + \mu_{t+1})/2] + \mu_{t-1}^2/2 + \mu_{t+1}^2/2}{2\,V/2}\right\} \\
& \propto \exp\left\{-\frac{\left\{\mu_t - [(\mu_{t-1} + \mu_{t+1})/2]\right\}^2}{2\,V/2}\right\}
\end{aligned}
$$

which is a Normal distribution:

$$p(\mu_t|\mu_{t-1}, \mu_{t+1}) = N\left(\frac{(\mu_{t-1} + \mu_{t+1})}{2}, \frac{V}{2}\right).$$

Therefore we can take it as the proposal density $g\left(\mu_t^\star|\mu_{1:T}^m\right)$ and help to simplify the calculation of the Metropolis ratio. Recall that Hastings (1970) proposed to

define the acceptance probability with

$$\alpha(\theta^\star, \theta) = min\left\{1, \frac{p(\theta^\star)g(\theta|\theta_\star)}{p(\theta)g(\theta^\star|\theta)}\right\}. \tag{4.6}$$

These values $\mu_t^\star$ and $\theta^\star$ are generated values based on the proposal distribution. In our case, to set the value at iteration $m+1$, we can simplify the test ratio as

$$\begin{aligned}
\frac{p(\theta^\star)}{g(\theta^\star|\theta)} &= \frac{p(\theta^\star|y_{1:T})}{g(\theta^\star|\theta_{t-1})} \\
&= \frac{p\Big(\mu_t^\star|y_{1:T}, \mu_{1:T}^m, \xi, \sigma, V\Big)}{g\Big(\mu_t^\star|\mu_{1:T}^m\Big)} \\
&= \frac{p(y_t|\mu_t^\star, \xi, \sigma) \cdot p(\mu_t^\star|\mu_{t-1}^m, \mu_{t+1}^m)}{p(\mu_t^\star|\mu_{t-1}^m, \mu_{t+1}^m)} \\
&= p(y_t|\mu_t^\star, \xi, \sigma).
\end{aligned}$$

And the accept/reject ratio is simplified as:

$$\begin{aligned}
r &= \frac{p(\theta^\star|y)/g(\theta^\star|\theta^{t-1})}{p(\theta^{t-1}|y)/g(\theta^{t-1}|\theta^\star)} \\
&= \frac{p(y_t|\mu_t^\star, \xi, \sigma)}{p(y_t|\mu_t^m, \xi, \sigma)}
\end{aligned}$$

which depends on two likelihood evaluations based on the GEV distribution. The MCMC posterior simulation is summarized as follows:

1. Set the initial values for all states and static parameters as discussed before.
2. For iteration $m = 1, 2...M$:
   - For time $t = 1, 2, ...T$,
     Draw $\mu_t^{m+1}|y_{1:T}, \mu_{t-1}^{m+1}, \mu_t^m, \mu_{t+1}^m, \sigma^m, \xi^m, V^m$ with a M-H step.
   - Draw $\sigma^{m+1}|y_{1:T}, \{\mu_t^{m+1}\}_{t=1}^T, \sigma^m, \xi^m$ with a M-H step.
   - Draw $\xi^{m+1}|y_{1:T}, \{\mu_t^{m+1}\}_{t=1}^T, \sigma^{m+1}, \xi^m$ with a M-H step.
   - Draw $V^{m+1}|y_{1:T}, \{\mu_t^{m+1}\}_{t=1}^T, \sigma^{m+1}, \xi^{m+1}, V^m$ with a M-H step.

To achieve faster convergence, we limit the possible range of the scale parameter within (0,30). Another reason to limit the range of the scale parameter is this can help to get reasonable estimation of parameters. The simulation study shows that without the limit on the scale parameter, the posterior estimation could be far from the true parameter ranges. The range of the shape parameter is limited within (-0.5, 0.5) to satisfy the regularity conditions of MLEs for the GEV distribution (Smith 1985; Ailliot, et al. 2011). To make the comparison consistent across all methods, I will use the same range limit for the scale and shape parameters in all particle filter methods.

We can extend the Metropolis-Hastings algorithm to a RW2 evolution for the states μ_t. The only difference with the RW1 evolution is the full conditional distributions of μ_t and the corresponding proposal density.

$$\begin{aligned} p(\mu_t|y_{1:T}, \{\mu_{-t}\}, \xi, \sigma, V) \propto\ & p(y_t|\mu_t, \xi, \sigma) \times p(\mu_t|\mu_{t-1}, \mu_{t-2}, V) \\ & \times p(\mu_{t+1}|\mu_t, \mu_{t-1}, V) \times p(\mu_{t+2}|\mu_{t+1}, \mu_t, V) \end{aligned}$$

and

$$\begin{aligned} & p(\mu_t|\mu_{t-1}, \mu_{t-2}, V) \times p(\mu_{t+1}|\mu_t, \mu_{t-1}, V) \times p(\mu_{t+2}|\mu_{t+1}, \mu_{t+1}, \mu_t, V) \\ & \propto \exp\left\{-\frac{[\mu_t - (2\mu_{t-1} - \mu_{t-2})]^2}{2\,V}\right\} \times \exp\left\{-\frac{[\mu_{t+1} - (2\mu_t - \mu_{t-1})]^2}{2\,V}\right\} \\ & \quad \times \exp\left\{-\frac{[\mu_{t+2} - (2\mu_{t+1} - \mu_t)]^2}{2\,V}\right\} \\ & \propto \exp\left\{-\frac{6\mu_t^2 - \mu_t[-2\mu_{t+2} + 8\mu_{t+1} + 8\mu_{t-1} - 2\mu_{t-2}]}{2\,V}\right\} \\ & \propto \exp\left\{-\frac{\mu_t^2 - 2\mu_t[-2\mu_{t+2} + 8\mu_{t+1} + 8\mu_{t-1} - 2\mu_{t-2}]/12}{2\,V/6}\right\} \\ & \propto \exp\left\{-\frac{\left\{\mu_t - [(-2\mu_{t+2} + 8\mu_{t+1} + 8\mu_{t-1} - 2\mu_{t-2})/12]\right\}^2}{2\,V/6}\right\}. \end{aligned}$$

This is a Normal distribution,

$$p(\mu_t|\mu_{t-1}, \mu_{t+1}) = N\left(\frac{(-2\mu_{t+2} + 8\mu_{t+1} + 8\mu_{t-1} - 2\mu_{t-2})}{12}, \frac{V}{6}\right)$$

Similar as with μ_t for a RW1 evolution model, I take this Normal probability as the proposal density and the accept/reject ratio is simplified as:

$$r = \frac{p(y_t|\mu_t^\star, \xi, \sigma)}{p(y_t|\mu_t^m, \xi, \sigma)}.$$

It is important to monitor the acceptance rate (the fraction of candidate draws that are accepted) of the Metropolis-Hastings algorithm. A too high acceptance rate may indicate the chain is probably not mixing well (not moving around the parameter space quickly enough). A too low acceptance rate may indicate the algorithm is too inefficient (rejecting too many candidate draws).

4.2.3 MCMC Diagnostics: Is the MCMC Method Working?

We check trace plots of the MCMC output to make sure they show no pattern and have roughly a horizontal line with no ascending/descending structure. We also look at the autocorrelation between samples. The MCMC samples have a tendency to be autocorrelated and are not independent. This should be examined empirically and the chain must be thinned if necessary.

The diagnostics result from one of the simulation study is illustrated here to demonstrate that the MCMC result achieve convergence. As shown in Figure 4.3. Plot (a) illustrate the trace plot of estimations of the scale parameter σ. Plot (b) illustrate the trace plot of estimations of the shape parameter ξ. Both plots show no pattern of pattern and is roughly horizontal, which illustrate the chains have settled down nicely between 5000 and 50000 iterations. Plots (c) and (d) are the autocorrelation function (ACF) plot of the estimates of σ and ξ respectively. The ACF rapidly approaches zero and stays there, which indicate no correlation between adjacent picked samples.

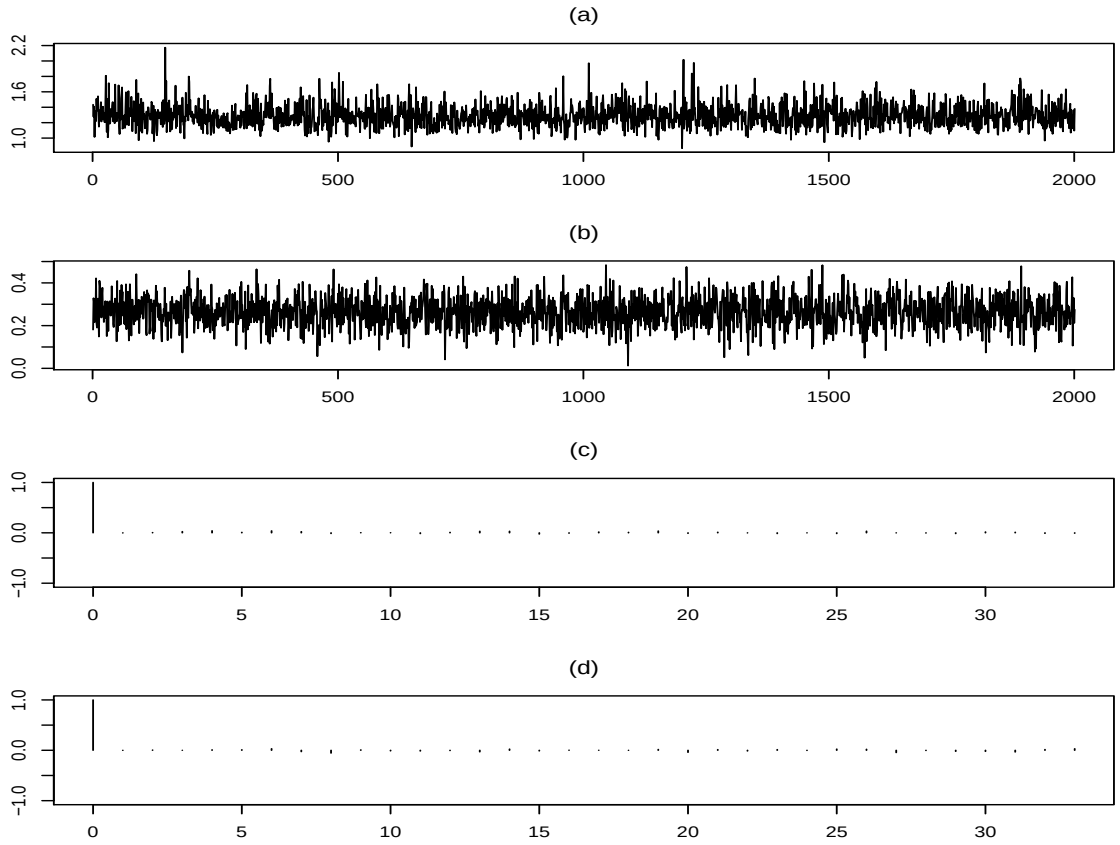

Figure 4.3: Diagnostics for the estimation of a dynamic GEV model. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

4.3 Particle Filters on Dynamic GEV Distribution

Notice that Storvik (Storvik 2002) and PL (Carvalho et al. 2010) particle filter methods require sufficient statistics to implement the algorithm; However, it is impossible to get finite dimensions sufficient statistics for the parameters of the GEV distribution. In this section, I will focus on the BS, APF and LW particle

filters only for one dynamic GEV model. Similarly as what we did in the first order DLM simulation study, I include the static parameters σ and ξ as part of the state vector for the BS and APF particle filters with added artificial evolution noise.

4.3.1 BS Filtering

1. Set initial values of μ_t, σ, ξ and V as in the MCMC case.

2. For time $t = 1, ..., T$,

   For each particle index $j = 1, ..., N$,

   i. Draw new samples from the dynamic model.
   Define a jittering dynamic variance: $V_\sigma = 0.9V_\sigma$ and $V_\xi = 0.9V_\xi$ and then propagate as

   $$\begin{aligned} \mu_t^j &\sim N(\mu_{t-1}^j, V_\mu), \\ log\sigma_t^j &\sim N(log(\sigma_{t-1}^j), V_\sigma), \\ \xi_t^j &\sim N(\xi_{t-1}^j, V_\xi). \end{aligned}$$

   ii. Calculate the weight:

   $$\omega_t^j \propto p(y_t|\mu_t^j, \sigma_t^j, \xi_t^j) \tag{4.7}$$

   given by $p(y_t|\mu_t^j, \sigma_t^j, \xi_t^j) = GEV(y_t|\mu_t^j, \sigma_t^j, \xi_t^j)$, the pdf of the GEV distribution.
   Compute and report the effective sample size $M_{T,eff} = \frac{1}{\sum_{j=1}^N (\omega_T^j)^2}$ when $t = T$.

   iii. Re-sample $\mu_t^j, \sigma_t^j, \xi_t^j$ based on the new weights.

4.3.2 APF Filtering

1. Set initial values as in the MCMC case.

2. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   Define a jittering dynamic variance: $V_\sigma = 0.9V_\sigma$ and $V_\xi = 0.9V_\xi$.

   i. Sample an auxiliary index vector $k^{(j)}, j = 1, ..., N$ by sampling the set $\{1 : N\}$ with the following weights:

   $$\mathbf{w}_t^j \propto p(y_t|\mu_{t-1}^j, \sigma_{t-1}^j, \xi_{t-1}^j)\omega_{t-1}^j \tag{4.8}$$

   with $p(y_t|\mu_{t-1}^j, \sigma_{t-1}^j, \xi_{t-1}^j) = GEV(y_t|\mu_{t-1}^j, \sigma_{t-1}^j, \xi_{t-1}^j)$, the pdf of the GEV distribution.

   ii. Propagate the new particles from these re-sampled particles

   $$\begin{aligned} \mu_t^j &\sim N(\mu_{t-1}^{k^{(j)}}, V_\mu), \\ log(\sigma_t^j) &\sim N\left(log(\sigma_{t-1}^{k^{(j)}}), V_\sigma\right), \\ \xi_t^j &\sim N(\xi_{t-1}^{k^{(j)}}, V_\xi). \end{aligned}$$

   where $k^{(j)}$ is index value of the j^{th} element of auxiliary index vector k generated at last phase.

   iii. Compute new weights using

   $$\omega_t^j \propto p(y_t|\mu_t^j, \sigma_t^j, \xi_t^j)/\mathbf{w}_t^{k^{(j)}} \tag{4.9}$$

   with $p(y_t|\mu_t^j, \sigma_t^j, \xi_t^j) = GEV(y_t|\mu_t^j, \sigma_t^j, \xi_t^j)$, the pdf of the GEV distribution.

   Compute and report the effective sample size $M_{T,eff} = \frac{1}{\sum_{j=1}^N (\omega_T^j)^2}$ when $t = T$.

   iv. Re-sample $\mu_t^j, \sigma_t^j, \xi_t^j$ based on the new weights.

4.3.3 LW Filtering

1. Set initial values as in the MCMC case.

2. For time $t = 1, ..., T$

   For each particle index $j = 1, ..., N$,

   i. Compute the sample covariance particles at time $t-1$:

$$\begin{aligned}
\mu_{log(\sigma)} &= \frac{1}{N}\sum_{i=1}^{N} log(\sigma_{t-1}^{i}), \\
var_{log(\sigma)} &= \frac{1}{N-1}\sum_{i=1}^{N}\left(log(\sigma_{t-1}^{i}) - \mu_{log(\sigma)}\right)^2 \\
\mu_{\xi} &= \frac{1}{N}\sum_{i=1}^{N}\xi_{t-1}^{i} \\
var_{\xi} &= \frac{1}{N-1}\sum_{i=1}^{N}\left(\xi_{t-1}^{i} - \mu_{\xi}\right)^2 \\
cov_{[log(\sigma),\xi]} &= \frac{1}{N-1}\sum_{i=1}^{N}\left(log(\sigma_{t-1}^{i}) - \mu_{log(\sigma)}\right)\left(\xi_{t-1}^{i} - \mu_{\xi}\right)
\end{aligned}$$

   Therefore, the covariance matrix is:

$$vpar_{t-1} = \begin{bmatrix} var_{log(\sigma)} & cov_{[log(\sigma),\xi]} \\ cov_{[log(\sigma),\xi]} & var_{\xi} \end{bmatrix}. \tag{4.10}$$

   The prior mean estimate is the expected (mean) value of the evolution density whose mean value is taken from particles at time $t-1$:

$$\begin{aligned}
U_t^j &= E(\mu|\mu_{t-1}^j) = \mu_{t-1}^j \\
m_{t-1}^j &= a(log(\sigma_{t-1}^j), \xi_{t-1}^j)^T + (1-a)\sum_{i=1}^{N}(log(\sigma_{t-1}^j), \xi_{t-1}^j)^T/N
\end{aligned}$$

   ii. Compute the first stage weights

$$\begin{aligned}
\mathbf{w}_t^j &\propto p(y_t|U_t^j, m_{t-1}^j) \\
&= p(y_t|\mu_{t-1}^j, \sigma_m^j, \xi_m^j)
\end{aligned}$$

$$= \quad GEV(y_t|\mu_{t-1}^j, \sigma_m^j, \xi_m^j)$$

where σ_m^j and ξ_m^j are the scale and shape parameter value taken from the m_{t-1}^j vector. An index vector k is generated by sampling on the set $\{1:N\}$ with these weights $\mathbf{w}_t^j$.

iii. Do a re-sampling on the scale and shape parameters with:

$$(log(\sigma_t^j), \xi_t^j)^T \sim N\left(m_{t-1}^{k^{(j)}}, (1-a^2)vpar_{t-1}\right) \tag{4.11}$$

and propagate the location μ_t with:

$$\mu_t^j \quad \sim N(\mu_{t-1}^{k^{(j)}}, V_\mu).$$

iv. Compute the second stage weights

$$\omega_t^j \propto p(y_t|\mu_t^j, \sigma_t^j, \xi_t^j)/\mathbf{w}_t^{k^{(j)}}. \tag{4.12}$$

Compute and report the effective sample size $M_{T,eff} = \frac{1}{\sum_{j=1}^N (\omega_T^j)^2}$ when $t = T$.

v. Perform re-sampling of μ_t^j, σ^j and ξ^j based on these new weights ω_t^j, i.e, pick a new index by sampling the set $\{1:N\}$ using weights generated from 4.12.

Chapter 5

Dynamic GEV Distribution Simulation Study

In this chapter, we analyse the performance of particle filters based on the study of simulated datasets for the dynamic GEV model. To simplify the process, we focus on the dynamic GEV model with static scale and shape parameters, while the location parameter μ_t is represented with a RW1 evolution with fixed variance V. Similarly as illustrated in Chapter 3 for the first order DLM simulation, all the simulations are based on observations that come in sequence. The dynamic GEV distribution is described by Equation 4.1. All the datasets are generated with fixed parameters $\sigma = 1.1$, $\xi = 0.1$ and μ_t is defined in the form of $\mu_t = \mu_{t-1} + \varepsilon_t$ with $\mu_0 = 510$ and evolution variance $V = 1$. We compare the approximation based on the MCMC and various particle filter methods by sequentially updating the filtering density $p(\mu_{1:t}, \sigma, \xi | y_{1:t}), t = 11, ..., T$ with new observations arriving. We start the comparison from $t = 11$ since smaller number of observations will not provide enough information to approximate the posterior density. For each t, the MCMC simulation is based on $M = 40,000$ iterations with a burn-in of 10,000 and a thinning equal to 30. The simulation study takes $N = 5000$ particles for

particle filter methods. To verify the performance of our algorithms, we start from a simpler case where some of the parameters in the model are assumed known.

5.1 Fixed Evolution Variance V of the Location Parameter

In this section, a simulation case to approximate the location μ_t, scale σ and shape ξ is studied when the evolution variance V of the location parameter is assumed known. The estimation result shows that the posterior means are close to the true values of the parameters and the true values are contained in the 95% credible intervals for the location and scale parameters. The results of the simulation are shown in Figures 5.1 to 5.5.

Figure 5.1 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the simulated observations from the dynamic GEV distribution. The solid line in red color are the true values of the location parameter μ_t. The filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line represents the posterior median values. This plot clearly shows the posterior median values of the MCMC method are close to the true μ_t parameter values. Plot (b) illustrates the comparison between MCMC and the BS particle filter approximation. Just like in plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line is used to represent the posterior median values. The solid line in blue color stands for the posterior median values computed from the BS particle filter approximation and the dashed lines in blue color cover the 95% credible interval. The solid black line representing the posterior median values of the MCMC method is not easily identified since it is overlapping with the solid blue line representing the posterior median values from

the particle filter method. The 95% credible interval from the BS particle filter also matches the MCMC result. Apparently, the BS particle filter achieves almost the same performance as the MCMC method to estimate the location parameter μ_t. Plot (c) summarizes the comparison between the MCMC and APF particle filter approximation. In a similar way, plot (d) illustrates the comparison between the MCMC and LW particle filter approximation. Plots (b), (c) and (d) show that the APF and LW particle filters achieve an homogeneous result to the MCMC method when estimating the state parameter. Figure 5.2 shows estimates of the posterior distribution for the scale parameter $p(\sigma|y_{1:t}), t = 11, ..., T$. In plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue color stands for the posterior median values of the BS particle filter approximation and the dashed lines in blue color give the 95% credible interval for the BS particle filter method. Evidently, the MCMC estimates provide smaller intervals than the BS particle filter estimates through all time points, $t = 11, ..., 200$. The posterior median values from both the MCMC and BS particle filter methods tend to agree with each other. Plot (b) illustrates the comparison between the MCMC and APF particle filter approximation. This plot clearly shows the APF particle filter achieves a similar result as the BS particle filter and tends to become consistent with the MCMC approximation for a large number of observations for both the posterior median values and the 95% credible interval. Plot (c) illustrates the comparison between the MCMC and LW particle filter approximation. The LW particle filter provides the smallest 95% credible interval among all these methods and is consistent with the MCMC result more closely compared to the BS and APF filters. Figure 5.3 shows estimates of the posterior distribution for the shape parameter $p(\xi|y_{1:t}), t = 11, ..., T$. In plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue stands for the posterior median

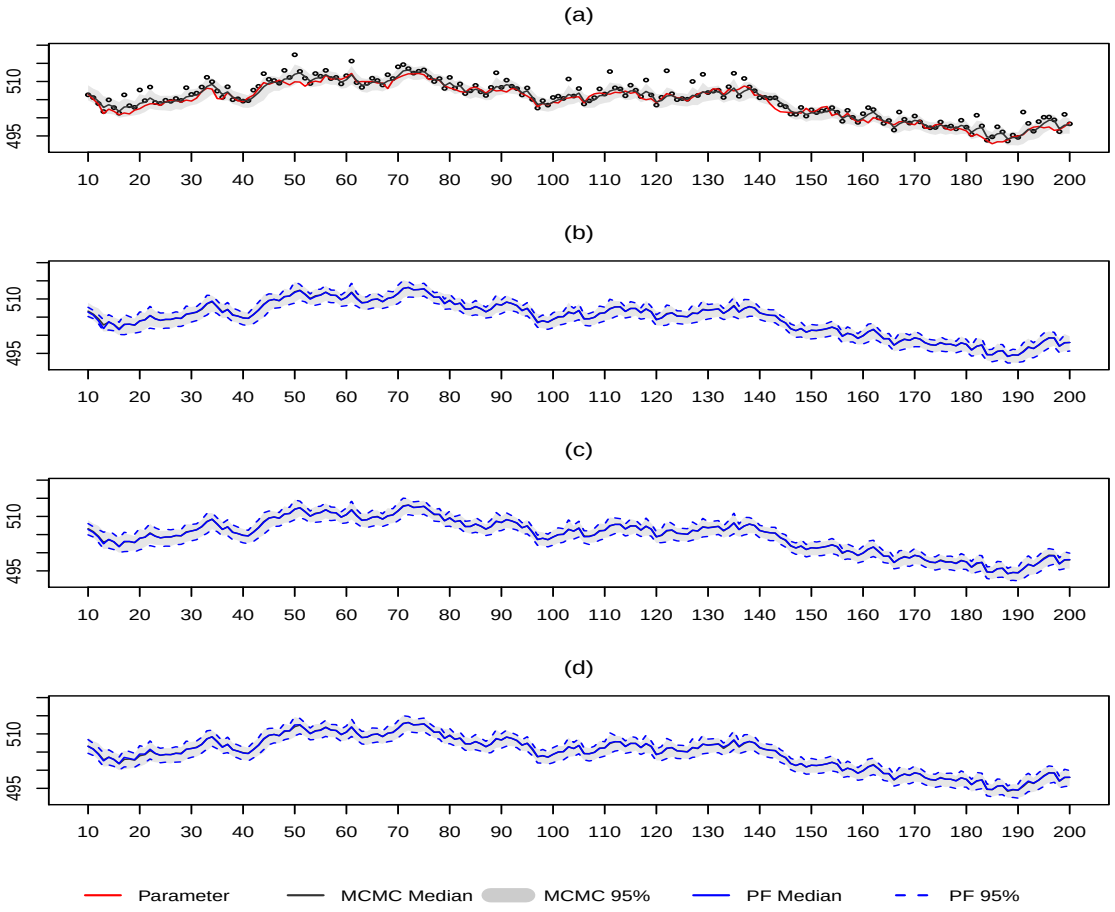

Figure 5.1: Estimation of the location parameter μ_t of a dynamic GEV model with known V. (a) True parameter vs MCMC, (b) MCMC vs BS, (c) MCMC vs APF, (d) MCMC vs LW.

values of the BS particle filter approximation and the dashed lines in blue cover the 95% credible interval. Apparently, the MCMC method has difficulty to approximate the true shape parameter value. Although the posterior median values of the approximation of the MCMC method are close to the true parameter value, 0.1. The 95% credible interval contains both positive and negative values and does not allow to make clear differences between the three types of extreme value distributions. The results of the BS particle filter show it has the same problem as

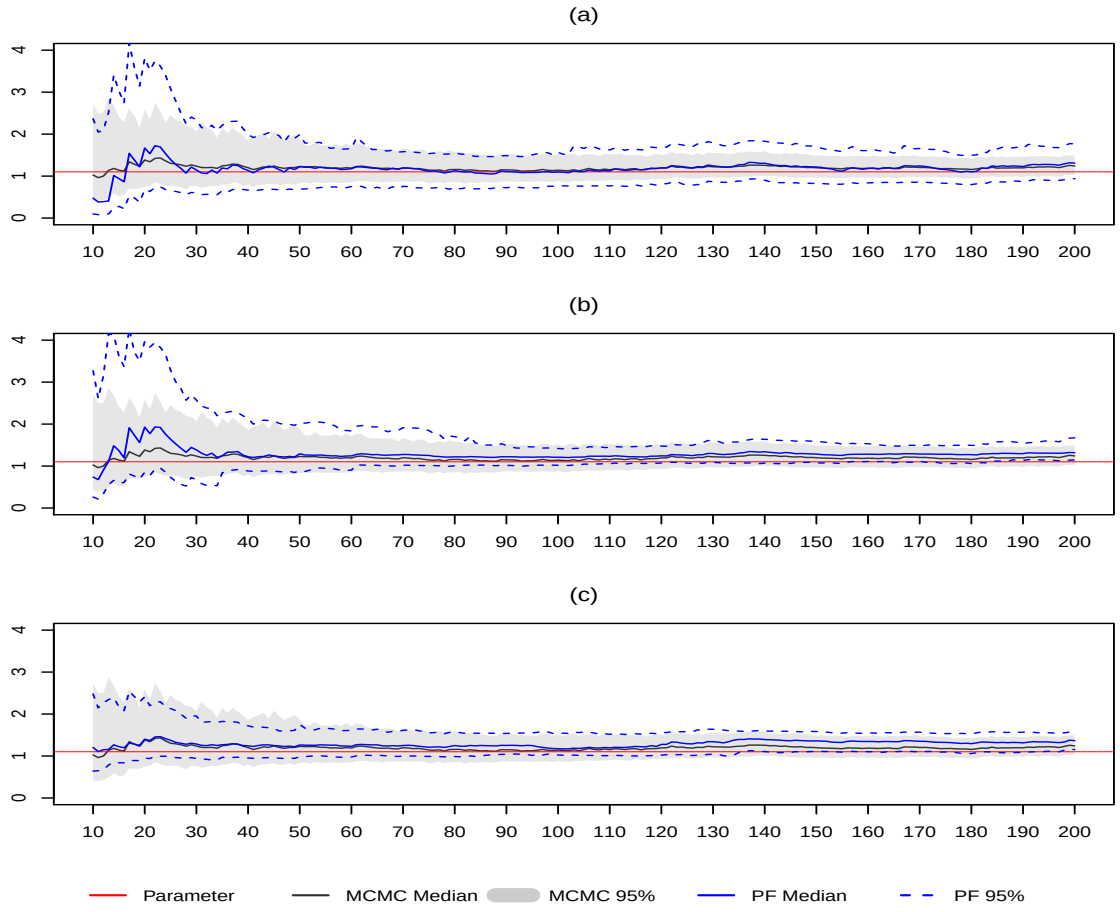

Figure 5.2: Estimation of the scale parameter σ of a dynamic GEV model with known V. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

the MCMC method. Plot (b) and plot (c) show comparisons of the MCMC with the APF and the LW particle filters approximation respectively. The 95% credible intervals of APF and LW particle filters provide values below the true parameter value 0.1 and move towards negative values. Apparently, all the particle filter methods have difficulty to estimate the shape parameter as well. Figure 5.4 illustrates some diagnostics of the methods based on the simulation. Plot (a) shows the M-H acceptance rate of the MCMC method for both the scale (in blue) and shape (in green) parameters. Both parameters have the acceptance rate within

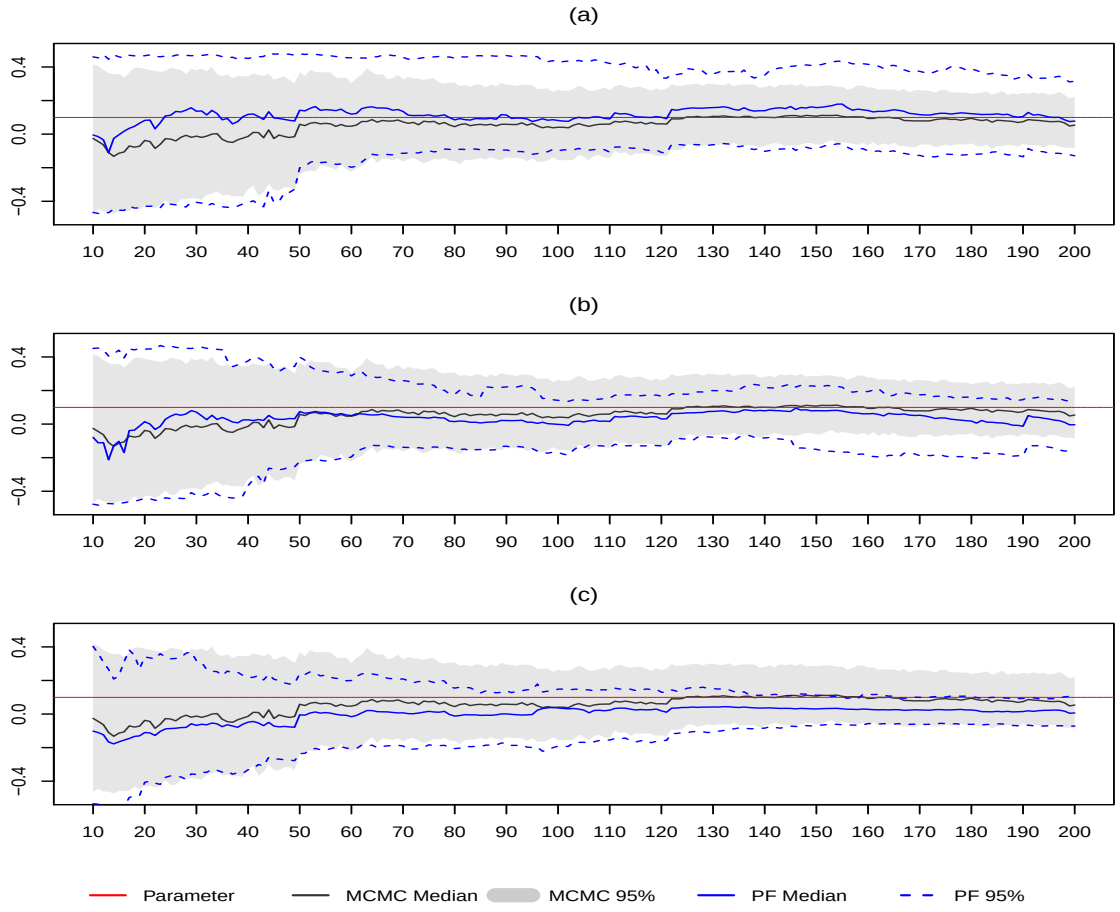

Figure 5.3: Estimation of the shape parameter ξ of a dynamic GEV model with known V. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

the range of 40% to 60% for number of observations is larger than 50. Plot (b) at the bottom shows the number of effective sample size from the three particle filter methods, with BS particle filter in blue, APF particle filter in green and LW particle filter in purple respectively. In this plot, there are sporadic drops in the effective sample size, usually around data outliers (see plot (a) of Figure 5.1) and break points, but they return to normal quickly afterwards, which indicates all the particle filter methods do not have the degeneracy issue discussed in section 2.1.2. Figure 5.5 shows histograms of samples from the marginal posterior approximation

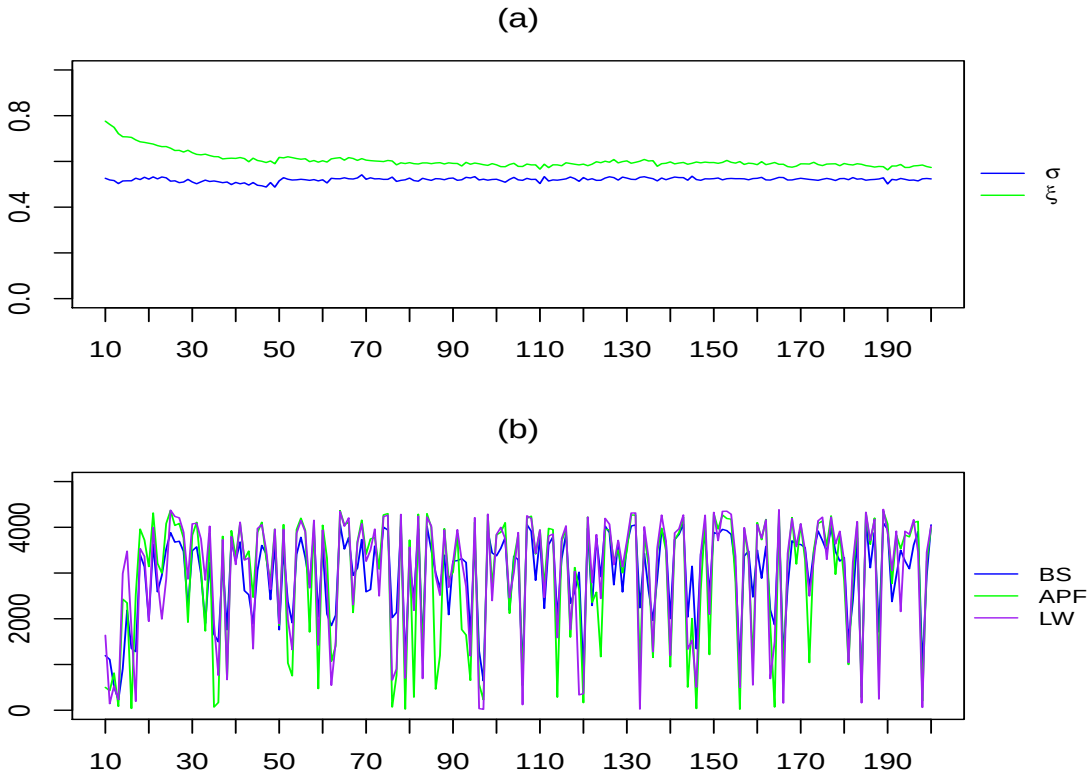

Figure 5.4: Diagnostics of the estimation of a dynamic GEV model with known V. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

taken at the last time point, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ for all the methods respectively. Histograms are limited to the same range in order to provide a clear comparison among the different methods. Histograms of the samples for $p(\mu_T|y_{1:T})$, in the left panel show all methods can approximate similarly the posterior distribution of the location parameter. In the center panel, the histograms of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ provide good estimation of the true parameter value, 1.1, across all methods. In the right panel, the histograms of the posterior distribution of the shape parameter $p(\xi|y_{1:T})$ spread

around negative and positive ranges, which make it impossible to distinguish the data from the three types of GEV distributions ($\xi > 0, \xi = 0, \xi < 0$).

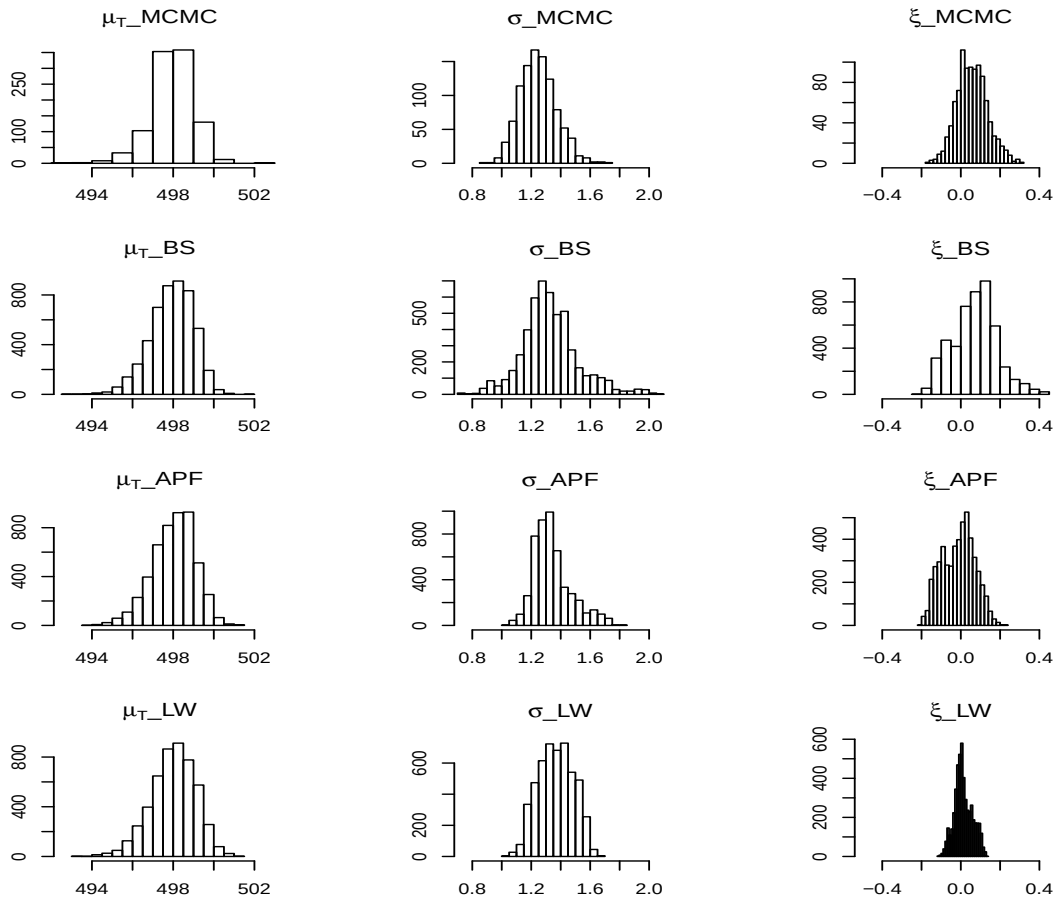

Figure 5.5: Histograms of posterior samples of a dynamic GEV model with known V. (Rows are by method, columns are by parameter.)

5.2 All Parameters are Unknown with $\xi = 0.1$

In this section, a simulation case to approximate the posterior distribution of all the model parameters is studied for the location μ_t, scale σ, shape ξ and the evo-

lution variance V of the location parameter. Under this scenario, the situation is such that all parameters are unknown and need to be approximated.

A preliminary study shows estimation of the scale and shape parameters is however, not stable after including the evolution variance V within the approximation. Instead, the posterior mean obtained from the MCMC method is used as a known evolution variance for particle filters. Therefore, I ran the MCMC algorithm with the first 50 observations and computed the posterior mean of the evolution variance $p(V|y_{1:50})$. Then plugged-in this posterior mean as the known V for particle filter estimations with new observations coming in. For the simulated dynamic GEV model, the parameters are set at $\sigma = 1$, $\xi = 0.1$, $V = 1$ and $\mu_t = \mu_{t-1} + \varepsilon_t$ with $\mu_0 = 510$. We compare the approximation from the MCMC and all particle filter methods by sequentially updating the filtering density $p(\mu_{1:t}|y_{1:t}), t = 11, ..., T$. The estimation result shows that all estimated posterior means are close to the true values and the true values are contained in the 95% credible intervals for the location and scale parameters. The results are shown in Figures 5.6 to 5.10.

Figure 5.6 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the simulated observations from the dynamic GEV distribution. The solid line in red stands for the true values of location parameter μ_t. The filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. This plot clearly shows the posterior median values from the MCMC method are close to the true μ_t values. Plot (b) illustrates the comparison between the MCMC and the BS particle filter approximation. Just like in plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line is used to represent the posterior median values. The solid line in blue color stands for the posterior median values of the BS particle filter approximation and the dashed lines in blue color define the 95% credible interval. In plot (b), the solid black line representing the posterior

median values from the MCMC method is not easily to be identified since it is overlapping with the solid blue line representing the posterior median values from the BS particle filter method. The 95% credible interval from the BS particle filter also matches with the MCMC results. Apparently, the BS particle filter achieves almost the same result as the MCMC method. Plot (c) illustrates the comparison between the MCMC and APF particle filter approximation. Plot (d) illustrates the comparison between the MCMC and LW particle filter approximation. Furthermore, plots (b), (c) and (d) prove that the APF and LW particle filters achieve similar results to the MCMC method. Figure 5.7 shows estimates of the posterior distribution for the scale parameter posterior distribution, $p(\sigma|y_{1:t}), t = 11, ..., T$. In plot (a), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line gives the posterior median values. The solid line in blue stands for the posterior median values of the BS particle filter approximation and the dashed lines in blue give the 95% credible interval. Evidently, the MCMC estimates provide a much larger interval than the BS particle filter estimates under a small number of observations (t less than 50). As the number of observations increases, the posterior median values and the 95% credible interval from both the MCMC and BS particle filter methods tend to agree with each other. Plot (b) illustrates the comparison between the MCMC and APF particle filter approximation. It clearly shows the APF particle filter achieves a similar result as the BS particle filter and tends to agree with the MCMC approximation for a large number of observations. Plot (c) illustrates the comparison between the MCMC and LW particle filter approximation. The LW particle filter provides the shortest 95% credible interval at small number of observations (less than 50) among all these methods. For a large number of observations, the LW particle filter agrees with the MCMC result but provides values slightly larger than the MCMC results. Figure 5.8 shows estimates of the posterior distribution for the shape parameter posterior distribution, $p(\xi|y_{1:t}), t = 11, ..., T$. In plot (a), the filled

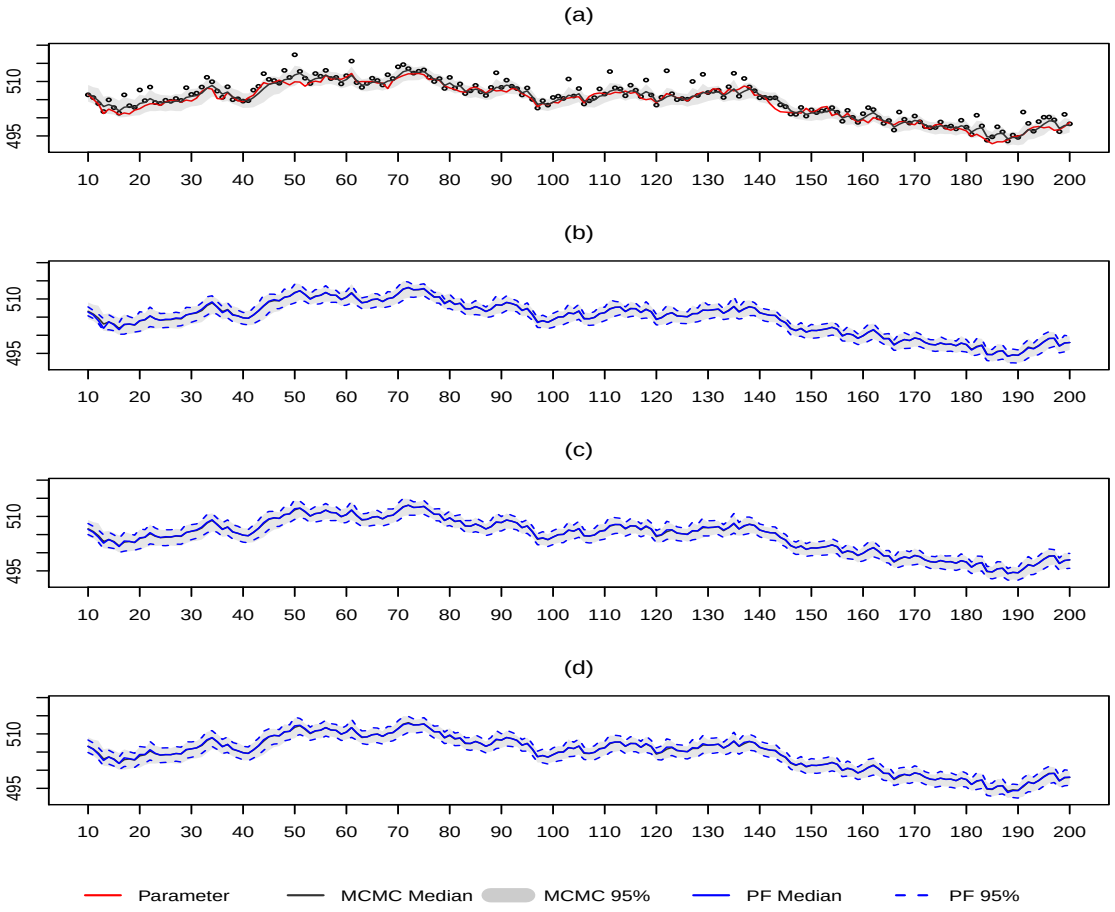

Figure 5.6: Estimation of the location parameter μ_t of a dynamic GEV model with all parameters unknown. (a) True parameter vs MCMC, (b) MCMC vs BS, (c) MCMC vs APF, (d) MCMC vs LW.

gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. Apparently, the MCMC method has difficulty to approximate the shape parameter. Although the posterior median values of the approximation of the MCMC method are close to the true parameter value, 0.1. The 95% credible interval contains both positive and negative values and is impossible to distinguish the three types of extreme value distributions. Also in plot (a), the solid line in blue stands for the posterior median

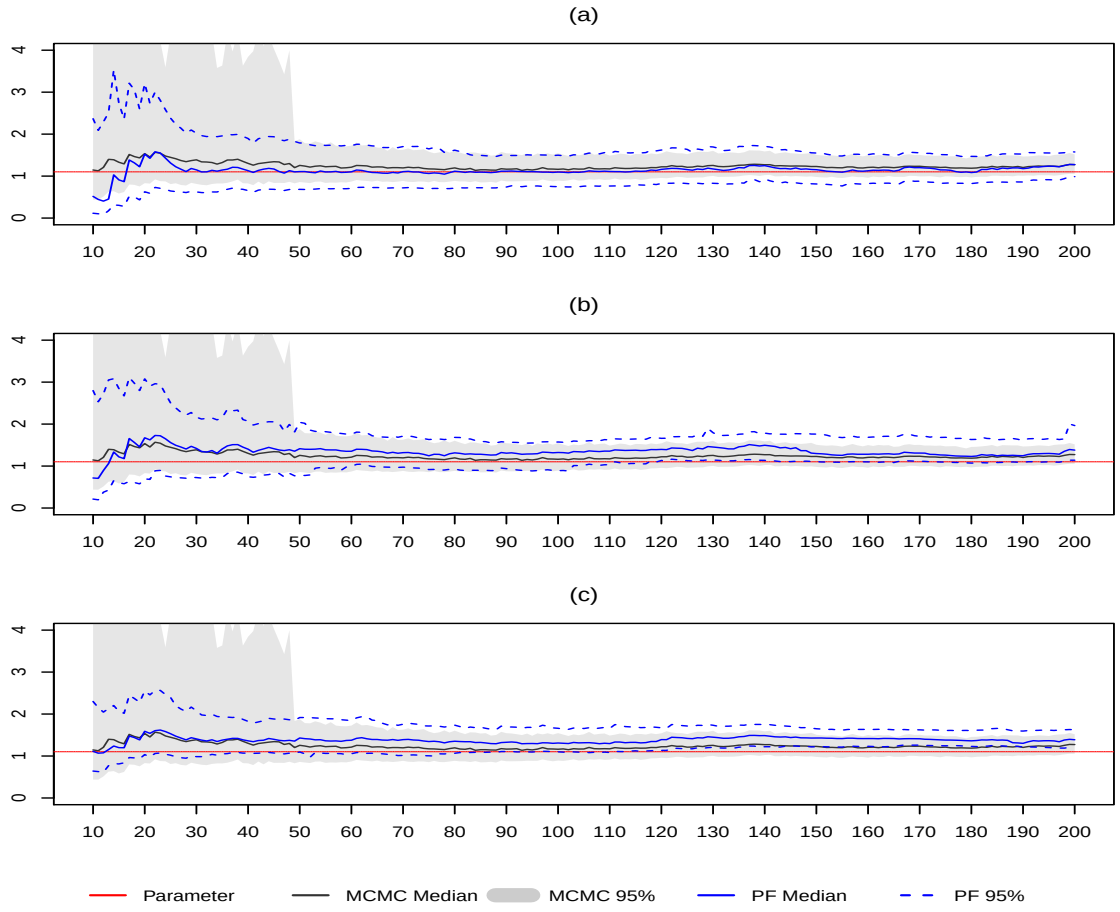

Figure 5.7: Estimation of the scale parameter σ of a dynamic GEV model with all parameters unknown. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

values of the BS particle filter approximation and the dashed lines in blue define the 95% credible interval. The approximation with the BS particle filter shows it has the same problem as the MCMC method. Plot (b) and plot (c) illustrate the comparisons of the MCMC with the APF and the LW particle filters approximation respectively. The 95% credible intervals of the APF and LW particle filters provide values below the true parameter value 0.1 and tend towards negative values. Apparently, all the particle filter methods have difficulty to approximate the

shape parameter as well. Figure 5.9 illustrates some diagnostics of the method-

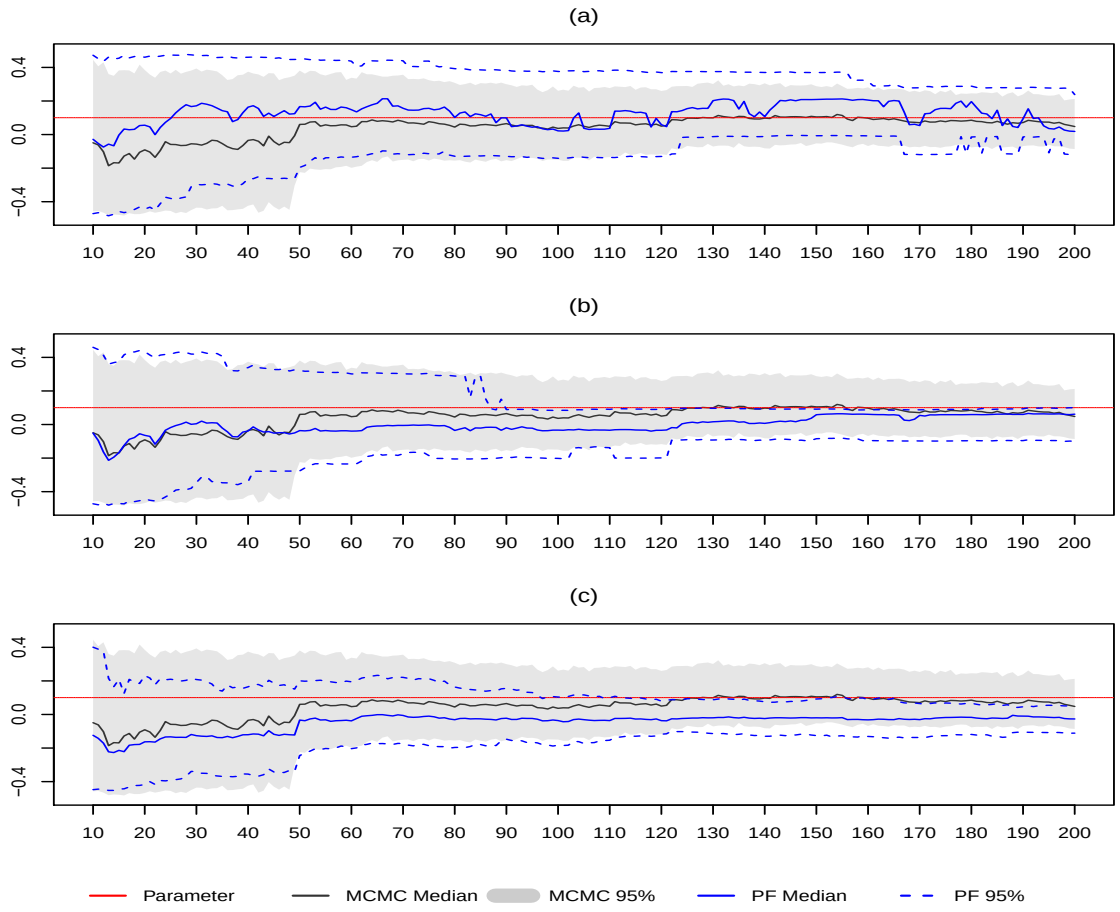

Figure 5.8: Estimation of the shape parameter ξ of a dynamic GEV model with all parameters unknown. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

ology, based on our simulation. Plot (a) shows the M-H acceptance rates of the MCMC method for both the scale (in blue) and shape (in green) parameters. Both parameters have the acceptance rate within the range of 40% to 60% when the number of observations is larger than 50. Plot (b) at the bottom shows the number of effective sample size for the three particle filter methods, with BS particle filter in blue color, APF particle filter in green color and LW particle filter in

purple respectively. In this plot, there are sporadic drops in the effective sample size, usually around data outliers (see plot (a) of Figure 5.6) and break points, but they return to stable quickly afterwards, suggesting that weight degeneracy is minor in this study. Figure 5.10 shows histograms of parameter approximation

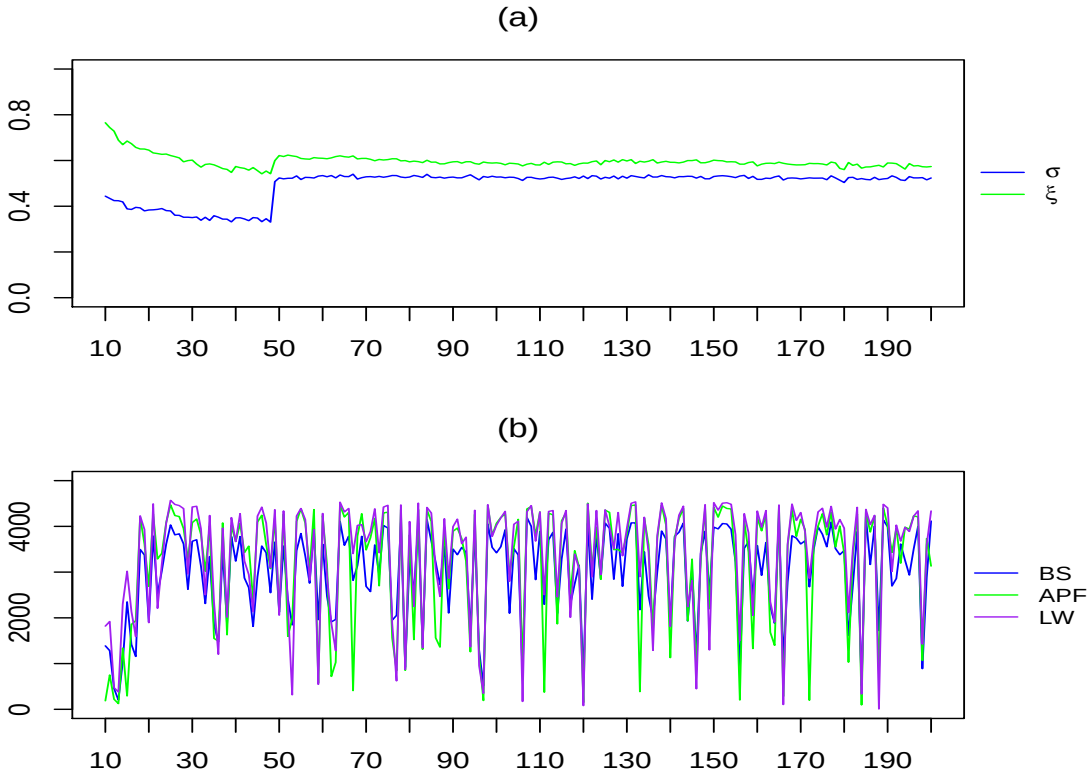

Figure 5.9: Diagnostics of the estimation of a dynamic GEV model with all parameters unknown. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

taken at the last time point of observations, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ of all methods respectively. Histograms are limited to the same range in order to provide a clear comparison among different methods. Histogram of samples for

approximation of the location parameter at the final time point, $p(\mu_T|y_{1:T})$, in the left panel show all methods can provide similar samples the location parameter. In the center panel, the histograms of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ provide good estimation of the true parameter value, 1.1, across all methods. In the right panel, the histograms of the posterior distribution of the shape parameter $p(\xi|y_{1:T})$ cover negative and positive values, which make it impossible to distinguish the three types of GEV distributions ($\xi > 0, \xi = 0, \xi < 0$). Compared to the previous case study where the given evolution variance V of the

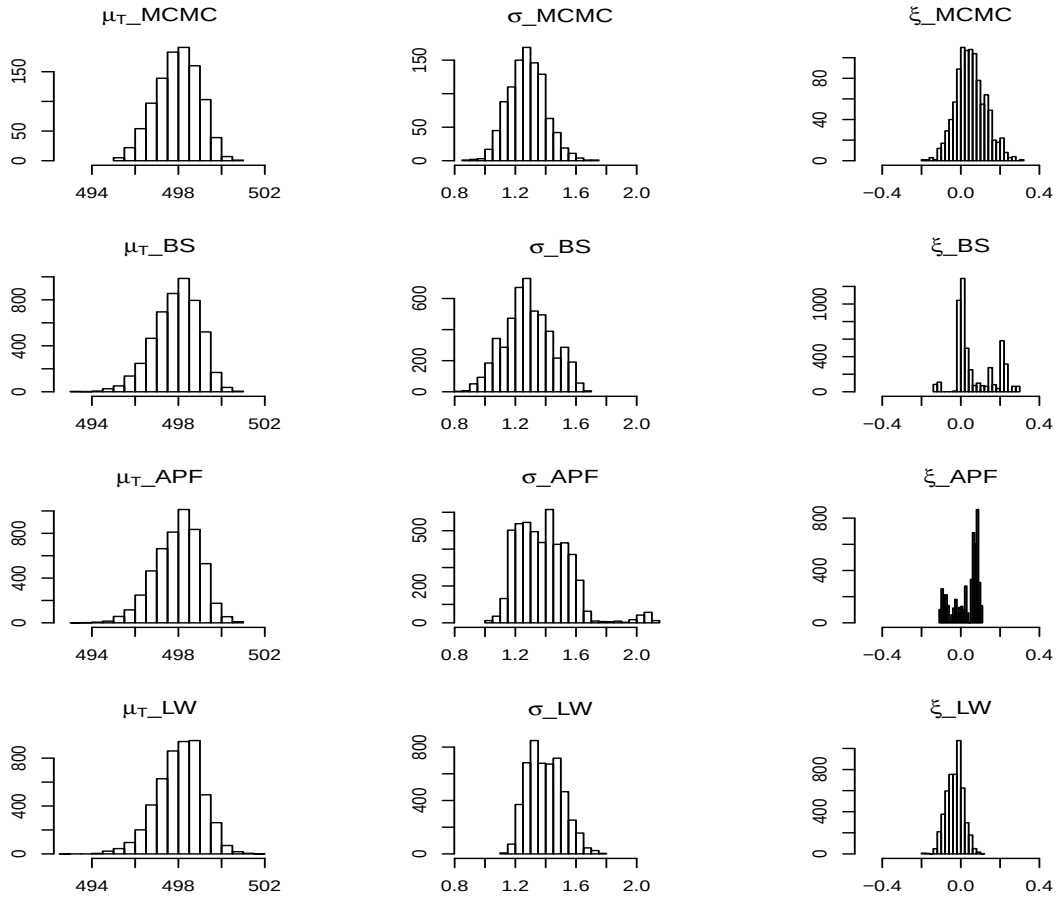

Figure 5.10: Histograms of posterior samples of a dynamic GEV model with all parameters unknown. (Rows are by method, columns are by parameter.)

location parameter is fully known, the approximations of this section shows no major difference to this other study. Another simulation study (not included here) also shows the approximation results are not so sensitive to the setting of the value of the evolution variance V. Hence, applying the posterior mean from the MCMC method as a true value for the particle filter study is acceptable.

5.3 All Parameters are Unknown with $\xi = 0.35$

The two dynamic GEV model cases we studied so far show no problem of estimating the state and scale parameters, but have difficulty to estimate the shape parameter when the value of the shape parameter ξ is close to 0. Pioneer study of the data sets that in the next chapter shows the shape parameter has mean value around 0.4, we want to check the particle filter performance on a larger value for the shape parameter. We generate a dataset using the same setting for the previous section but change the value of ξ from 0.1 to 0.35. Then we approximate all the parameters as in Section 5.2. The estimation result shows that all estimated posterior means are close to the true values of all the parameters including the shape parameter and the true parameter values are contained in the 95% credible intervals. The simulation result are shown in Figures 5.11 and 5.12. In this case, the results for the state and scale parameters is not shown since they are similar to the previous example.

Figure 5.11 shows estimates of the posterior distribution for the shape parameter posterior distribution, $p(\xi|y_{1:t}), t = 11, ..., T$. In plot (a), the filled gray area represents the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The 95% credible interval of the MCMC method begins to contain positive values only after the number of observations is larger than 100 and does contain the true shape parameter value, 0.35. The solid

line in blue is the posterior median values of the BS particle filter approximation and the dashed lines in blue provide the 95% credible interval. The 95% credible interval of the BS particle filter follows the same trend as the MCMC method, it starts to contain positive values at the same time point as the MCMC method does. The overall 95% credible interval from the BS particle filter has almost the same width span as the interval obtained from the MCMC method. Plot (b) and plot (c) compare the MCMC with the APF and the LW particle filters approximation respectively. The APF and LW particle filters follow a similar trend as the MCMC method, but with a smaller interval. For a large number of observations ($t > 100$), the posterior means from all methods are estimated to be positive and the 95% credible intervals do not contain zero. Figure 5.12 shows histograms from the posterior distributions at the last time point, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ of all methods respectively. These histograms are limited to the same range in order to provide a clear comparison among the different methods. Histograms of the samples from the location parameter at the final time point, $p(\mu_T|y_{1:T})$, in the left panel show all methods can approximate similarly the location parameter with similar median values and 95% credible intervals. In the center panel, the histograms of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ contain the true parameter value, 1.1, across all methods. Regarding the estimates of the shape parameter as shown in the right panel, the histograms of the posterior distribution of $p(\xi|y_{1:T})$ spread in positive ranges only, which make it possible to detect the data arising from a type Frèchet ($\xi > 0$) of the GEV family of distributions.

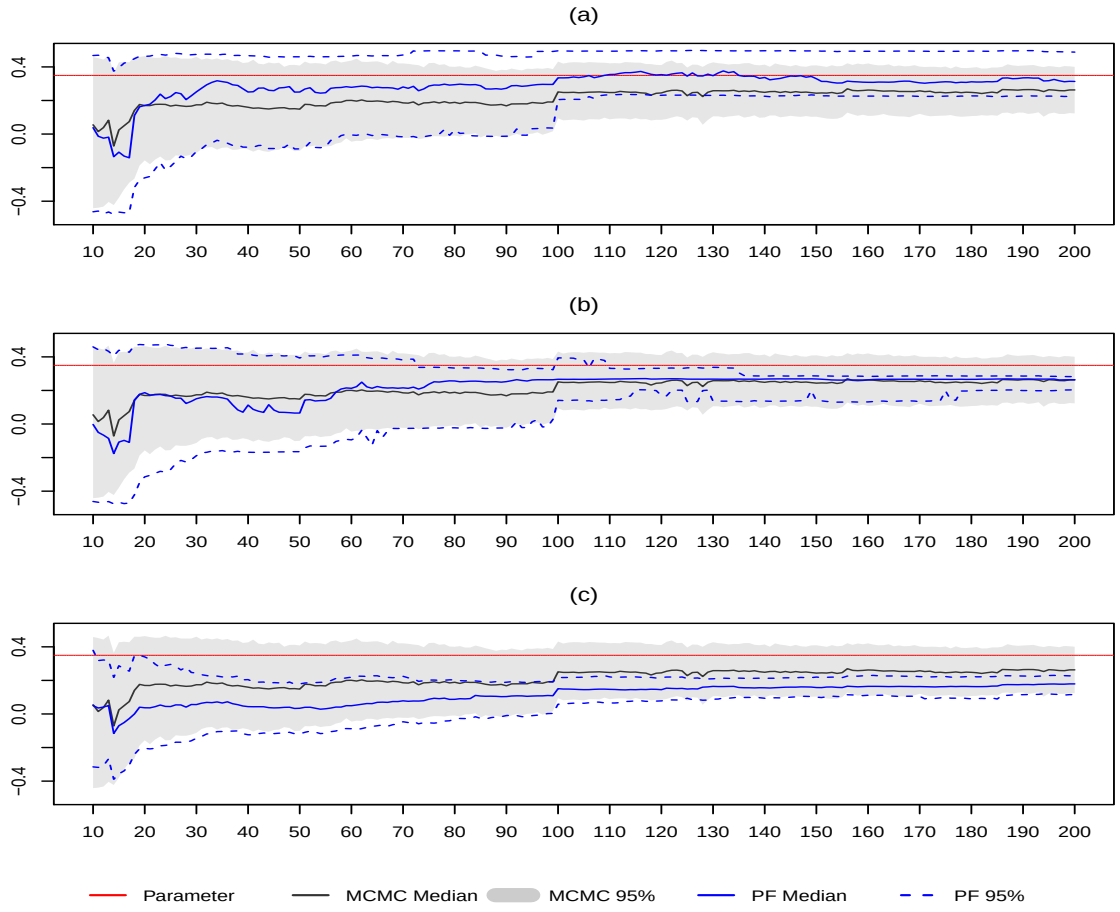

Figure 5.11: Estimation of the location parameter ξ of a dynamic GEV model with all parameters unknown (ξ=0.35). (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

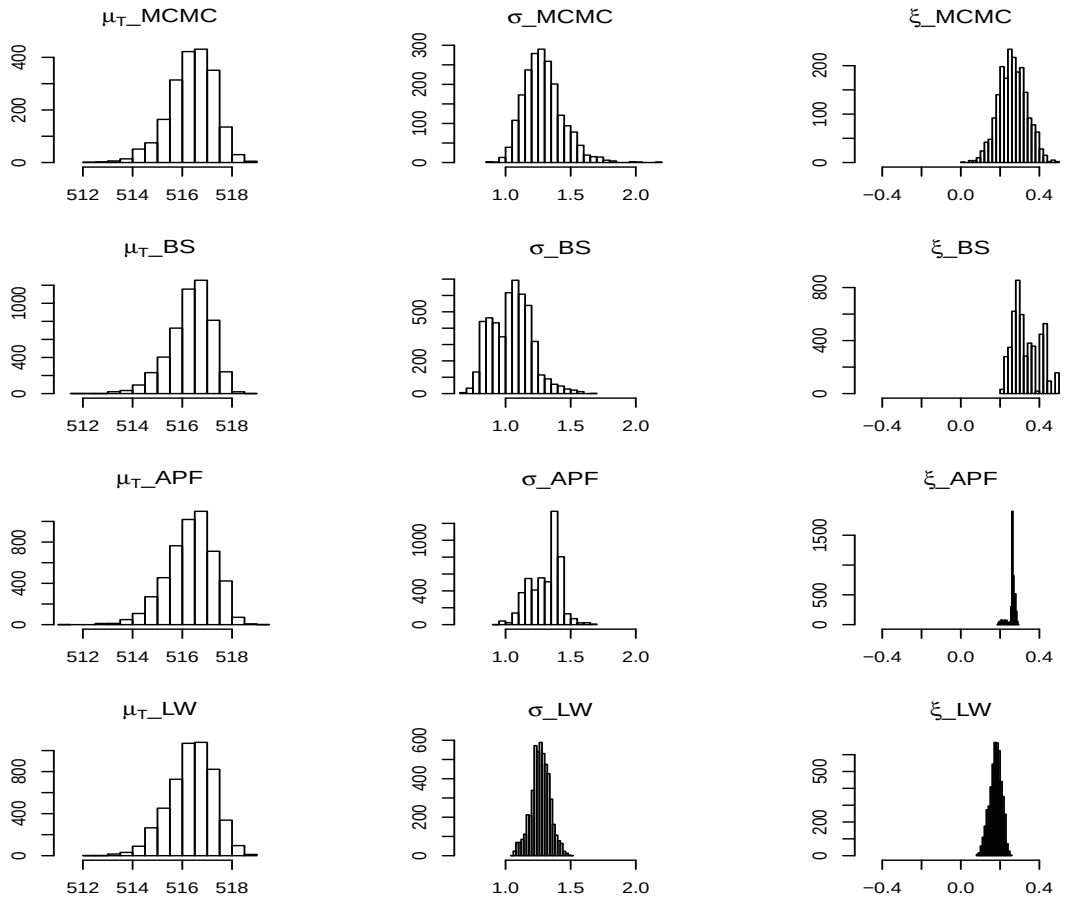

Figure 5.12: Histograms of posterior samples of a dynamic GEV model with all parameters unknown (ξ=0.35). (Rows are by method, columns are by parameter.)

5.4 All Parameters are Unknown with $\xi = -0.35$

The case in the previous section shows the MCMC method and all the particle filter methods can well approximate the dynamic GEV model when the value of the shape parameter set is around 0.4. Now I explore what is the result for the case when the shape parameter is set to a negative value that is relatively far from 0. A preliminary study shows estimates from the MCMC method are chaotic under a negative value of the shape parameter, so I decided to extend the number of observations to be $T = 500$ and work on a generated dataset with the same setting as the previous section, except the value of ξ is -0.35. Then, I approximate all the parameters similarly to what I did in Section 5.2 and 5.3. The estimation result shows that the MCMC method has difficulty to estimate either the scale or the shape parameter. On the other hand, all the particle filter methods do a good job to estimate all these parameters. The simulation results are shown in Figures 5.13 to 5.17.

Figure 5.13 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the simulated observations from the dynamic GEV distributions. The solid line in red stands for the true location parameter μ_t. The filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. This plot clearly shows that the posterior median values from the MCMC method are close to the true μ_t values. Plots (b), (c) and (d) illustrate the comparison between the MCMC with the particle filters approximation. The filled gray area represents the 95% credible interval of the MCMC simulation and the solid black line is used to represent the posterior median values. The solid line in blue stands for the posterior median values of the particle filter approximation and the dashed lines in blue color define the 95% credible interval. It can be seen that all the particle filter methods provide similar estimates as the MCMC

method but with a smaller credible interval. Figure 5.14 shows estimates of the

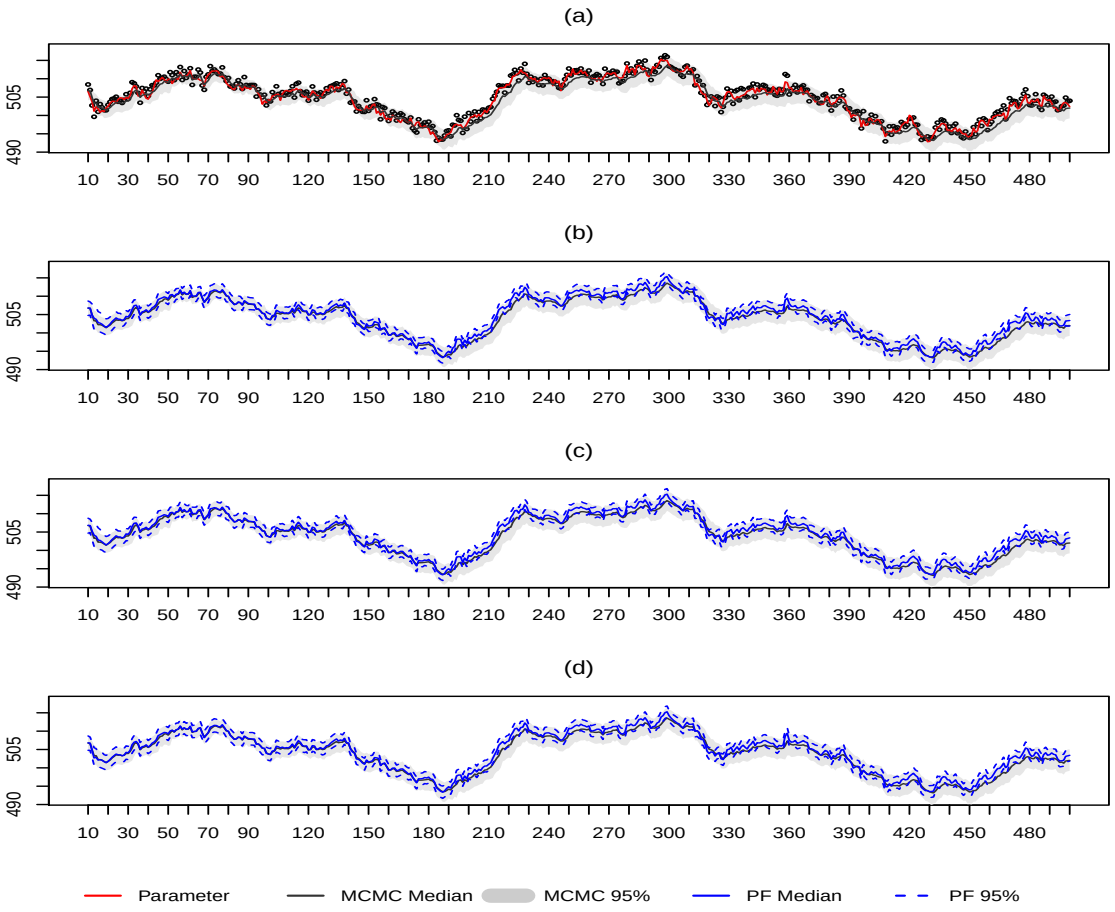

Figure 5.13: Estimation of the location parameter μ_t of a dynamic GEV model with all parameters unknown (ξ=-0.35). (a) True parameter vs MCMC, (b) MCMC vs BS, (c) MCMC vs APF, (d) MCMC vs LW.

posterior distribution for the scale parameter $p(\sigma|y_{1:t}), t = 11, ..., T$. In all three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue stands for the posterior median values of the particle filter approximation and the dashed lines in blue cover the 95% credible interval. Apparently, the MCMC method lost track of the posterior density of the scale parameter. On the

other hand, the 95% credible intervals from these particle filter methods contain the true scale parameter value and tend to be stable within a small span width. Figure 5.15 shows estimates of the posterior distribution for the shape parameter

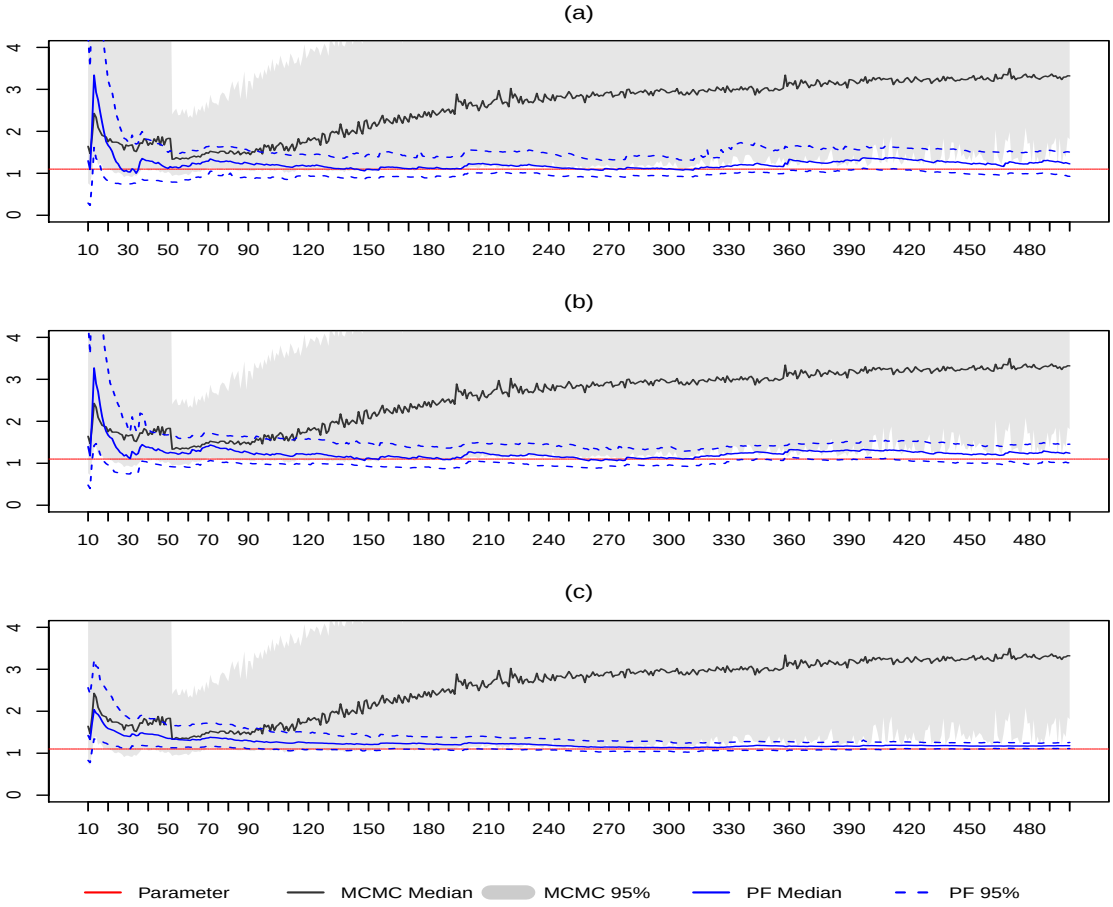

Figure 5.14: Estimation of the scale parameter σ of a dynamic GEV model with all parameters unknown (ξ=-0.35). (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

$p(\xi|y_{1:t}), t = 11, ..., T$. In all three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue stands for the posterior median values of the particle filter approximation and the dashed lines in blue provide a

95% credible interval. Evidently, the 95% credible intervals for all these methods contain the true shape parameter value and and define a negative range only. However, the posterior median values based on the MCMC method is much lower than the true parameter value and tends to go beyond the boundary value of -0.5. Figure 5.16 illustrates some diagnostics of the simulation. Plot (a) shows the M-H

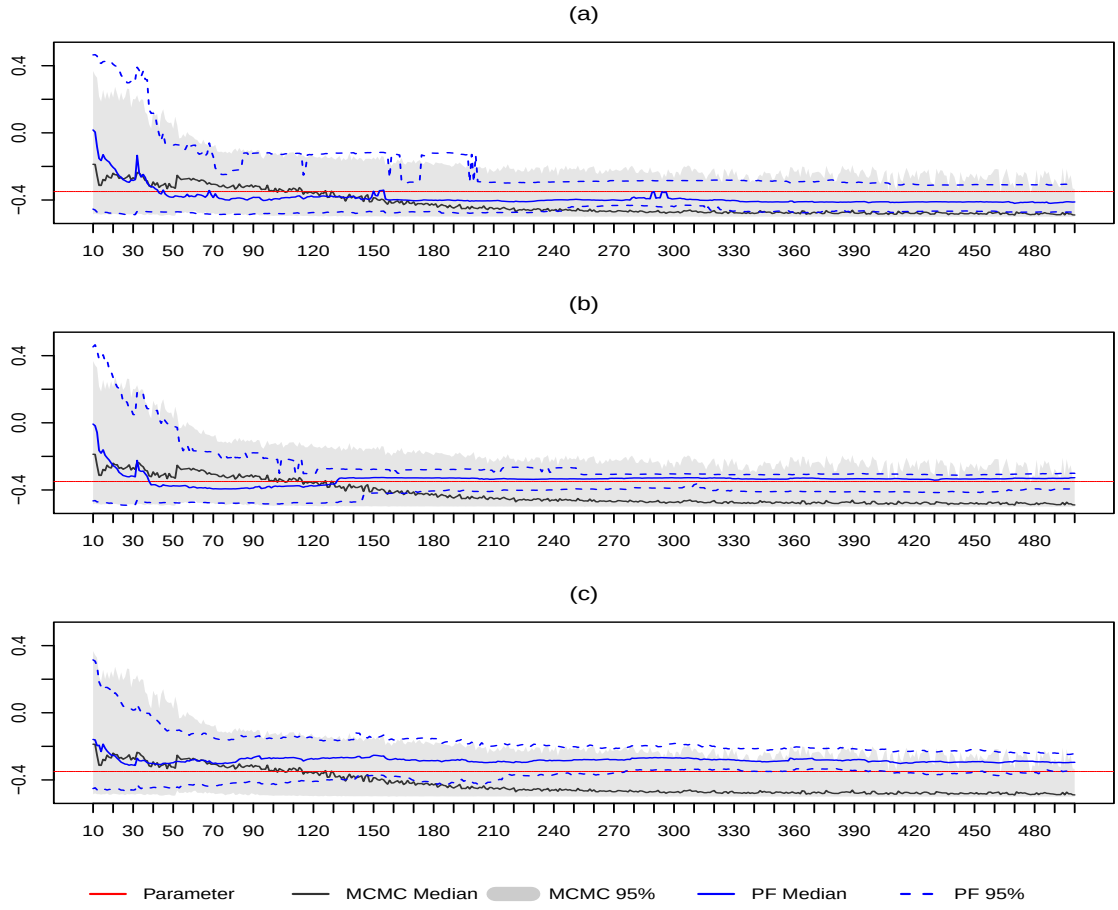

Figure 5.15: Estimation of the shape parameter ξ of a dynamic GEV model with all parameters unknown (ξ=-0.35). (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

acceptance rate of the MCMC method for both the scale (in blue) and shape (in green) parameters. Both parameters have an acceptance rate below 40% for most

time points. An enormous number of iterations is needed in order to achieve the MCMC method to get a higher acceptance rate. Plots (b), (c) and (d) show the number of effective sample size from the three particle filter methods respectively. Compared to the previous cases, these plots show more frequent sporadic drops in the effective sample size, which are around data outliers. Check plot (a) of Figure 5.13, we can see more outliers appear in this dataset. However, the drops in the effective sample size return to normal quickly afterwards, suggesting that weight degeneracy is not so much of a concern in this study. Figure 5.17 shows histograms of posterior samples of parameters considered at the last time point of observations T, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ of all methods respectively. Histograms describe the same parameter are limited to the same range in order to provide a clear comparison among different methods. Histograms of the posterior samples of the location parameter at the final time point, $p(\mu_T|y_{1:T})$, in the left panel show all methods can well approximate the location parameter, except that the approximation of the MCMC method cover a large range. In the center panel, the histograms of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ provide evidence that all particle filters are capable of estimating the true parameter value, while the MCMC method covers a large range and has difficulty to track the true parameter value. The right panel displays the histograms of samples from the posterior distribution of $p(\xi|y_{1:T})$. Basically for all methods, the posterior samples range within negative values, which makes it possible to distinguish a Weibull distribution. The simulation from the MCMC method is quite different than with particle filters since a large number of samples are around -0.5 or even smaller.

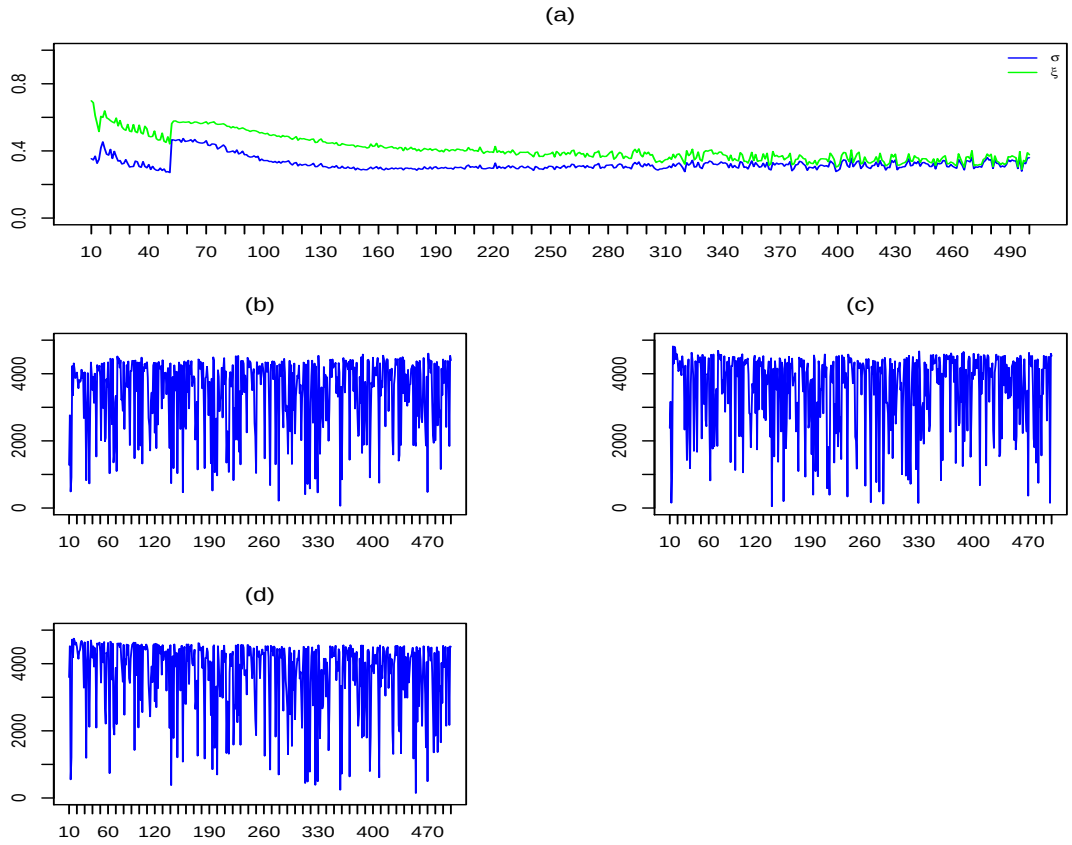

Figure 5.16: Diagnostics of the estimation of a dynamic GEV model with all parameters unknown (ξ=-0.35). (a) MCMC M-H acceptance rate, (b) P.F. effective sample size of BS Filter, (c) P.F. effective sample size of APF Filter, (d) P.F. effective sample size of LW Filter.

5.5 Simulation Summary

Several constraints were discovered during the simulation study in this chapter. A preliminary study suggests the evolution variance V of the location parameter needs to be limited in a small scale (as not larger than 3) in order to get an accurate approximation of the scale and shape parameters. Furthermore, the simulations

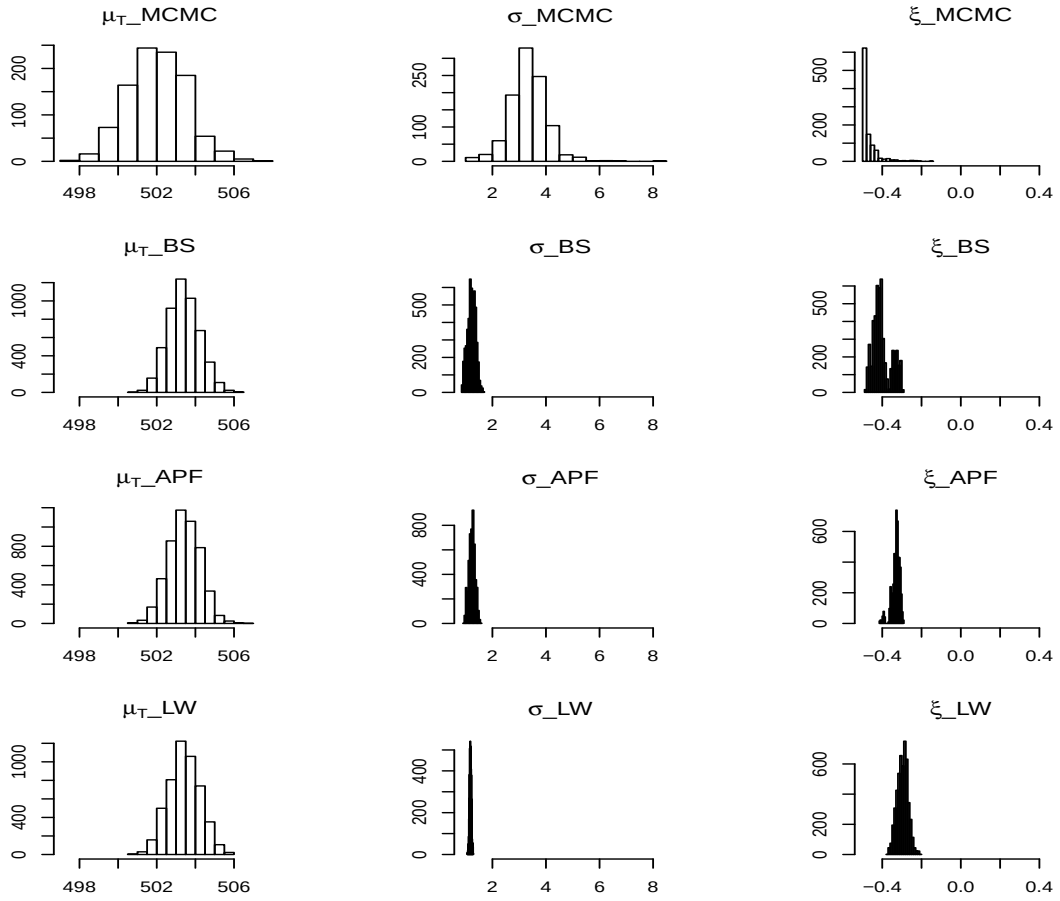

Figure 5.17: Histograms of posterior samples of a dynamic GEV model with all parameters unknown (ξ=-0.35). (Rows are by method, columns are by parameter.)

show that adding an artificial evolution noise of the scale and shape parameters in the BS and APF particle filters does help to approximate the posterior density of the scale and shape parameters respectively. However, the challenge is how to set the jittering of the artificial evolution variance. Trial and error simulation is used to detect a decrease factor between 0.8 and 0.99 over time as an optimal choice. Furthermore, during all simulation studies, a log transformation is employed for the scale parameter σ in order to propose a Gaussian evolution on this parameter. We can summarize the simulation results for the dynamic GEV model as follows:

1. For all the methods, including the MCMC and the three particle filter methods we studied in this chapter, the estimated posterior densities of the location parameter $p(\mu_t|y_{1:t})$ from particle filters are close to the MCMC location estimates. Basically the 95% credible interval from all methods can contain the true parameter values.

2. In most of our study cases, the approximation of the posterior density of the scale parameter of particle filters matches with the result from the MCMC method for relatively large number of observations, such as $t > 50$. Although the intervals are not exactly the same, the 95% credible intervals from the particle filters are a close match to the MCMC credible intervals.

3. All these methods have problems with the shape parameter estimation when the parameter value is close to 0. For a larger value of the shape parameter, the 95% credible intervals from the particle filters tend to provide a better estimation than MCMC methods.

Chapter 6

Data Applications with the Dynamic GEV Model

In this chapter, we analyse the performance of particle filter algorithms and compare it with the MCMC method through three data examples that had been studied by fitting a dynamic GEV model in several papers. The settings and steps to estimate the parameters of the dynamic GEV models are close to the procedures implemented in Section 5.2, 5.3 and 5.4. The posterior mean of V obtained from the MCMC method is used as a known evolution variance for particle filter methods. More specifically, we run the MCMC algorithm with part of the observations and compute the posterior mean of the evolution variance of $p(V|y_{1:t})$. Then this posterior mean is the known V for particle filter simulations with observations arising sequentially. As in the previous chapter, we start the comparison from $t = 11$ since a smaller number of observations will not provide enough information to approximate the posterior density.

6.1 Athletic Records Data Set

The first case study is for the annual world record for women's 3000m race from 1972 to 1993. Robinson and Tawn (1995) first studied the record to assess whether Wang Junxia's world record in 1993 was consistent with the previous data by fitting a GEV model. Their study was subsequently discussed by Smith (1997) and replied by Robinson and Tawn (1997). Gaetan and Grigoletto (2004) extended these analyses by using particle filter methods to model a dynamic trend by assuming that μ_t follows a second-order random walk trend while σ and ξ are kept as static parameters. Fearnhead, Wyncoll and Tawn (2010) extended their method by adding in more data from recent years.

Although the state estimation from Gaetan and Grigoletto (2004) looks attractive, the estimations of the scale and shape parameters are not provided in their paper. On the other hand, Fearnhead, Wyncoll and Tawn (2010) estimated σ and ξ by an EM algorithm and reported that the scale parameter σ is around 4 while the shape parameter ξ is around -0.1 for this dataset. As shown in Figures 6.1 to 6.5, my study achieves a similar result to theirs and so the algorithms presented in this thesis get further verified.

Figure 6.1 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the observations of the annual world record data. For all the three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line represents the posterior median values. The solid line in blue stands for the posterior median values of the particle filter approximation and the dashed lines in blue cover the 95% credible interval. Evidently, all three particle filters achieve a result that is comparable to the MCMC method. The dynamic GEV model we proposed seems a good candidate to deal suitably with the location parameter in a time varying RW1 model. Figure 6.2 shows estimates of the posterior distribu-

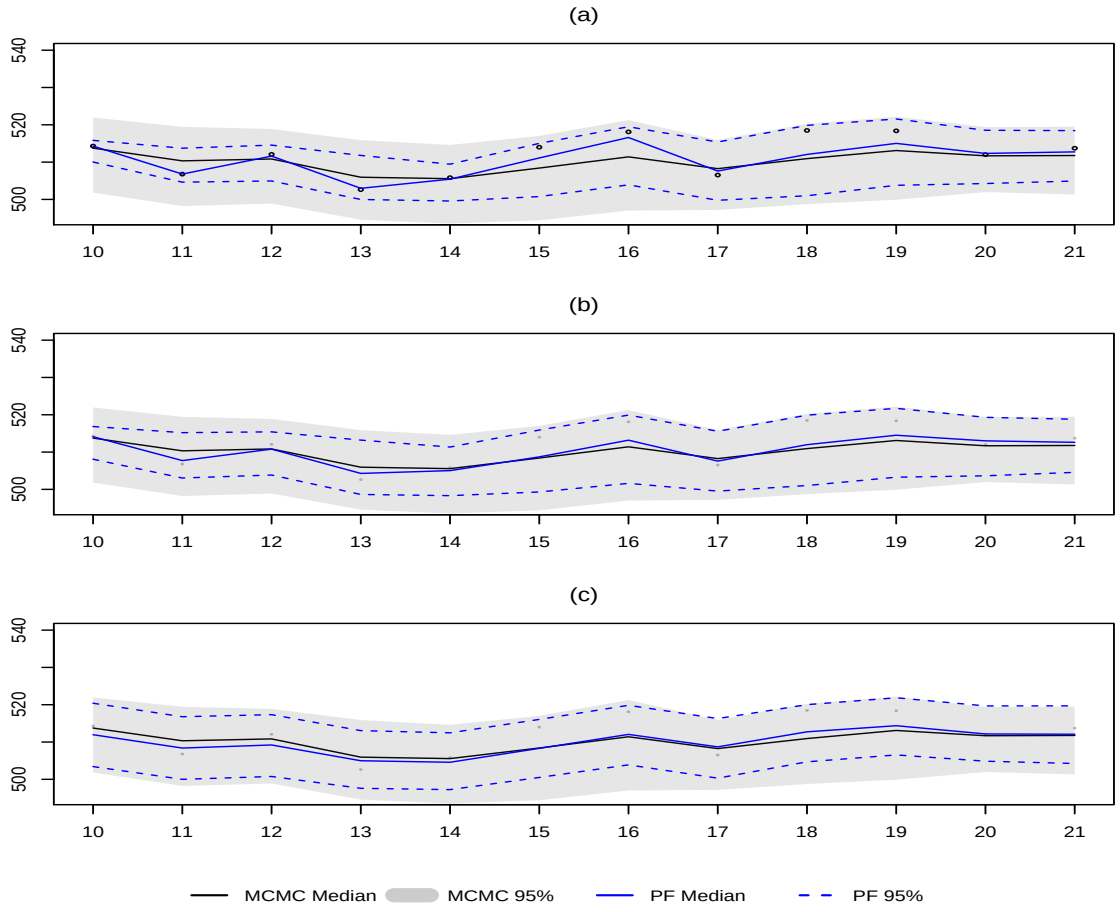

Figure 6.1: Estimation of the location parameter μ_t of the women's 3000m race data set using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

tion for the scale parameter $p(\sigma|y_{1:t}), t = 11, ..., T$. For all three plots, (a), (b) and (c), the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue stands for the posterior median values of the different particle filter approximation and the dashed lines in blue color provide the 95% credible interval for the particle filter methods. Since this data has a very limited number of observations ($T = 21$), a solid conclusion from the posterior density cannot be

drawn here. Overall, the LW particle filter provides a more stable estimation and matches with the MCMC method. The MCMC and LW particle filter show that the posterior density approximation of the scale parameter spans from 5 to 10 and the median value is around 7. This result agrees with the conclusion drawn by Fearnhead, Wyncoll and Tawn (2010). Figure 6.3 shows estimates of the posterior

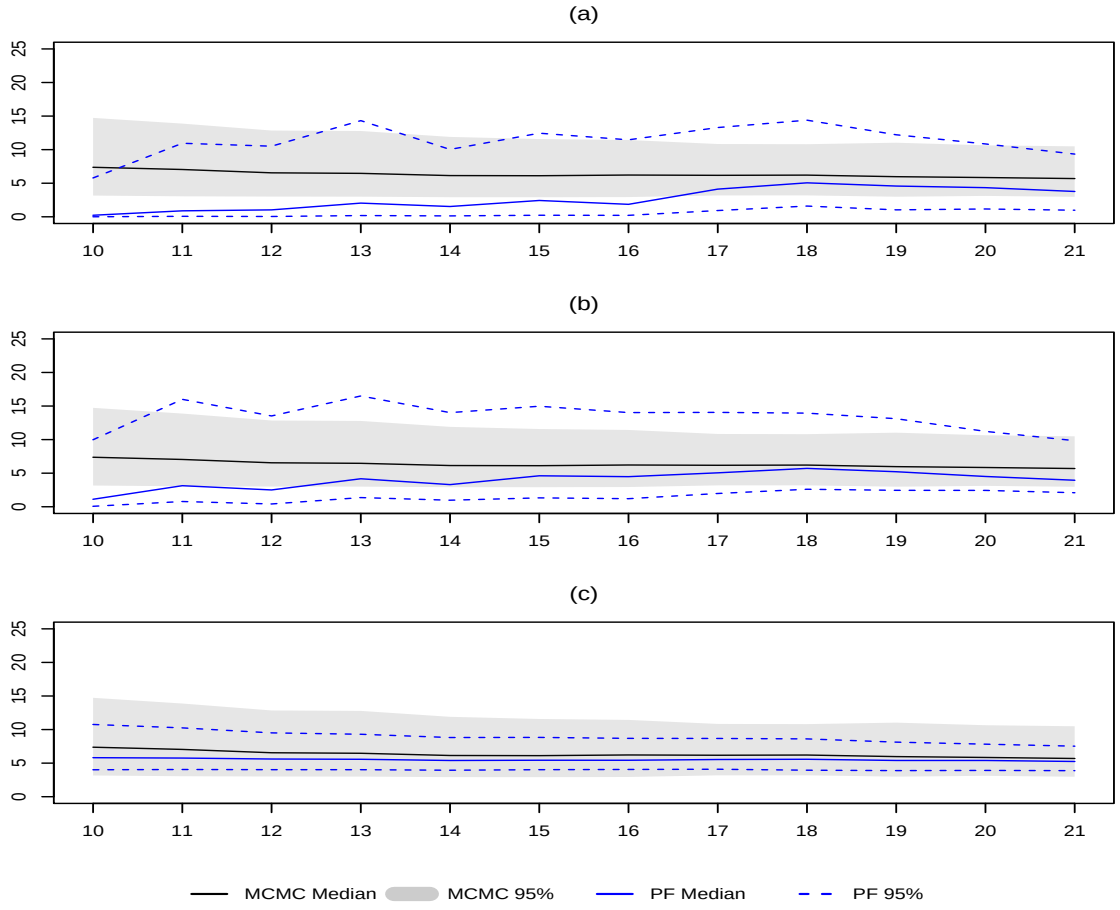

Figure 6.2: Estimation of the scale parameter σ of the women's 3000m race data set using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

distribution for the shape parameter $p(\xi|y_{1:t})$, $t = 11, ..., T$. In all three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation

and the solid black line stands for the posterior median values. The solid line in blue represents the posterior median values of the BS particle filter approximation and the dashed lines in blue cover the 95% credible interval for the particle filter methods. Apparently, all methods have difficulty to distinguish any of the three types of GEV distributions ($\xi > 0, \xi = 0, \xi < 0$). On the other hand, the posterior median values are close to -0.1 or -0.2. Again, this matches with the estimation result of Fearnhead, Wyncoll and Tawn (2010). Figure 6.4 illustrates

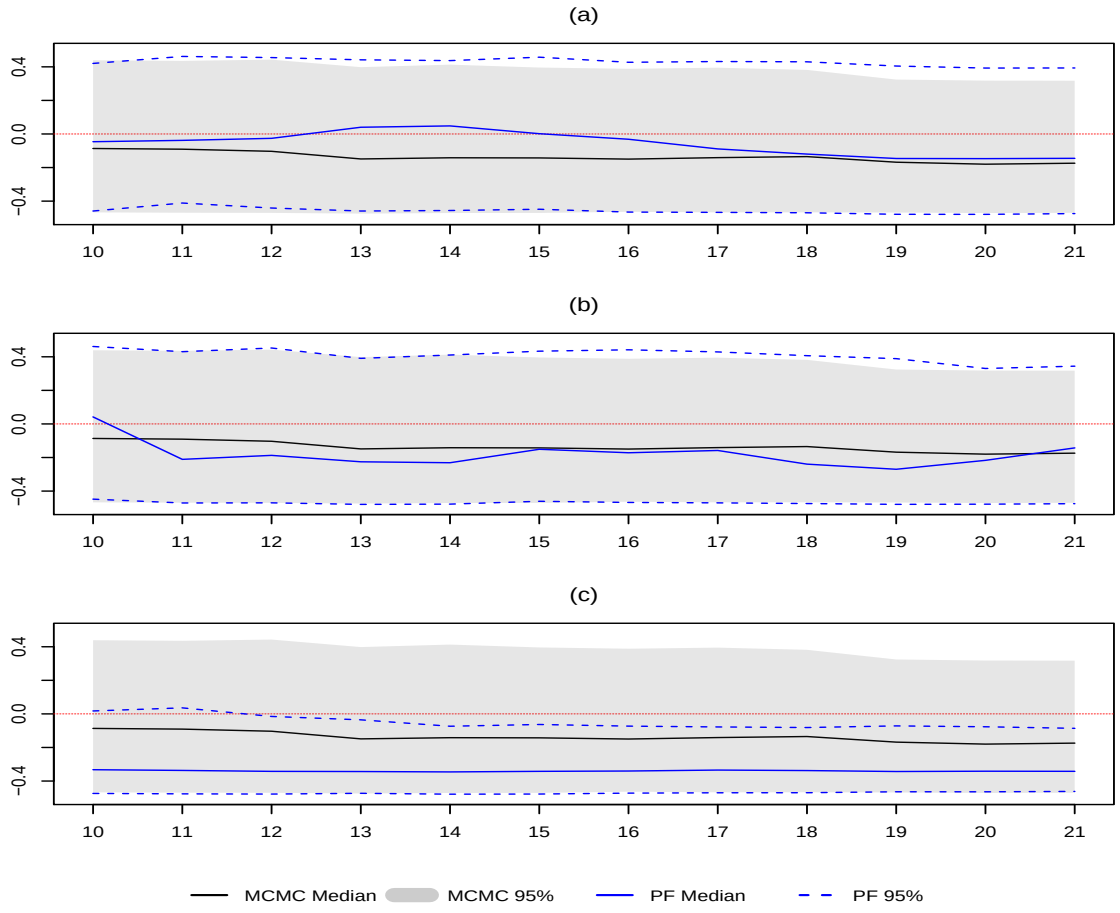

Figure 6.3: Estimation of the shape parameter ξ of the women's 3000m race data set using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

some diagnostics of the methods based on my estimation. Plot (a) shows the M-H acceptance rate of the MCMC method for both the scale (in blue) and shape (in green) parameters. The scale parameter σ has acceptance rate within the range of 40% to 60% and the shape parameter ξ has an acceptance rate larger than 60%. Plot (b) at the bottom shows the number of effective sample size from the three particle filter methods, with the BS particle filter in blue, the APF particle filter in green and the LW particle filter in purple respectively. Since the number of observations is very limited in this dataset, the effective sample size produced does not seem to be convincing enough. This plot shows the effective sample sizes from the BS and APF particle filters are low. On the other hand, the result from the LW particle filter has a larger number of the effective sample size compared to the BS and APF particle filters, suggesting that weight degeneracy from the LW particle filter method is lower in this study. Figure 6.5 shows histograms of samples from the marginal posterior approximation obtained at the last time point, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ of all the methods respectively. Histograms are limited to the same range in order to provide a clear comparison among the different methods. Histograms of the samples for $p(\mu_T|y_{1:T})$, in the left panel show all methods provide a similar range and median values. In the center panel, the histograms of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ also provide similar range and median values across all methods. In the right panel, the histograms of the samples of the posterior distribution approximation of the shape parameter $p(\xi|y_{1:T})$ spread around negative and positive ranges, which make it impossible to distinguish one of the three types of GEV distributions. However, it appears that the shape parameter tends to be negative as shown by the mode of the histograms.

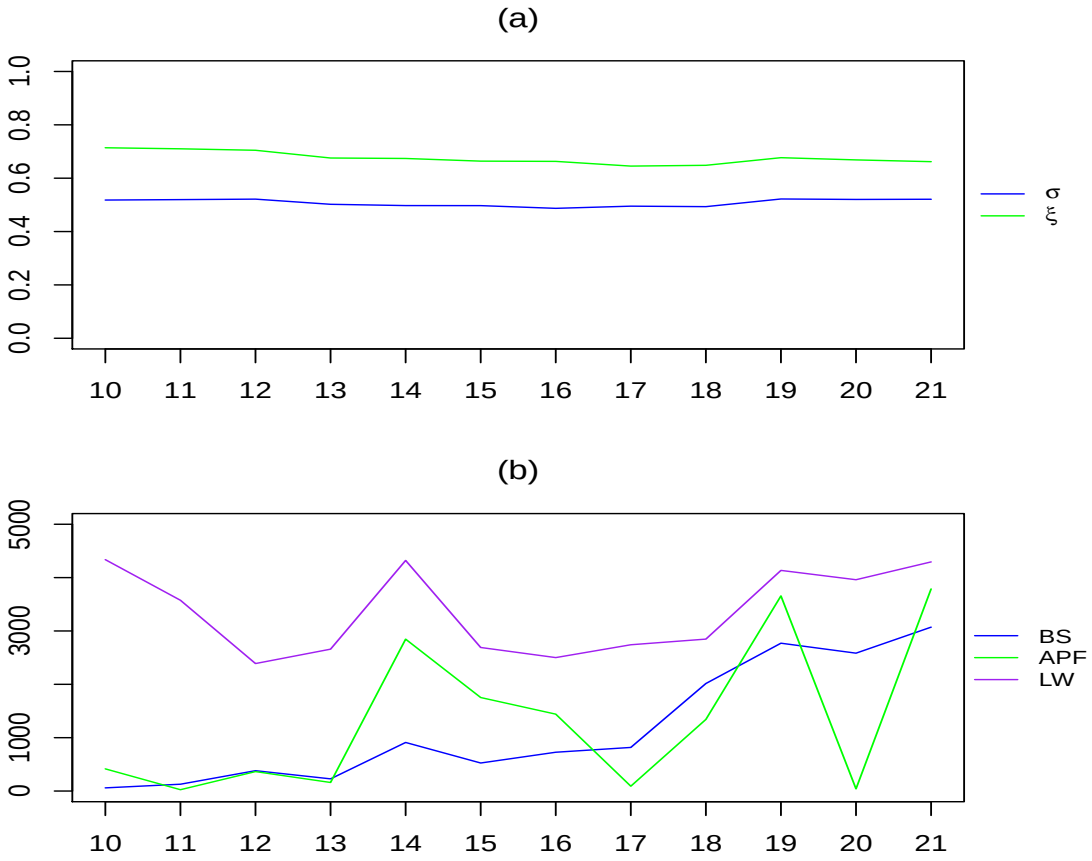

Figure 6.4: Diagnostics of the estimation of the women's 3000m race data set using a dynamic GEV model. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

6.2 Rainfall Data

In this example we study the maximum monthly rainfall values (in millimeter) from January 1961 to November 1999 taken at the Maiquetía station located at the Simón Bolívar Airport near Caracas, Venezuela. Huerta and Sansó (2007) first studied this data and which is further discussed by Huerta and Stark (2012). The result from their study using a MCMC method with fixed location parameter

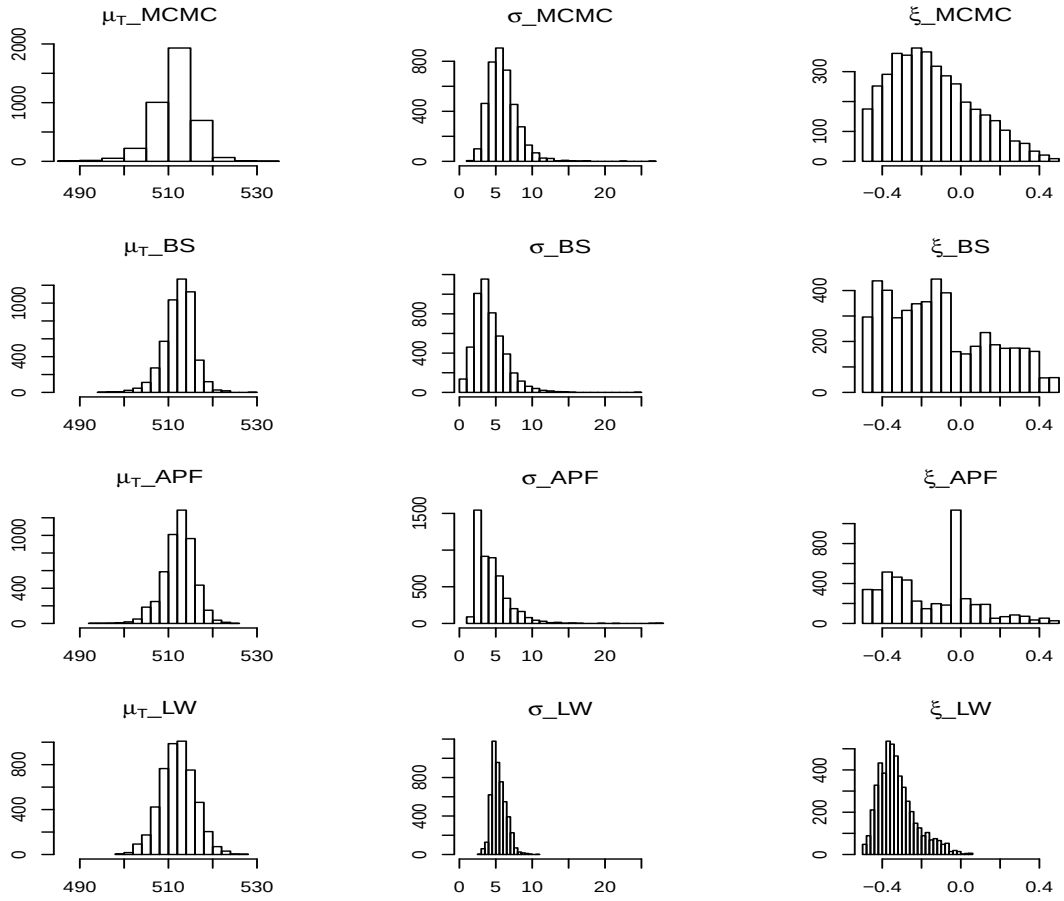

Figure 6.5: Histograms of posterior samples of the women's 3000m race data set using a dynamic GEV model. (Rows are by method, columns are by parameter.)

shows the scale parameter is between 8 to 11 for a 95% credible interval with median value around 9.5. They also concluded that the shape parameter is between 0.3 to 0.7 with median value around 0.5. In Chapter 4, we discussed to limit the value of the shape parameter within (-0.5, 0.5) to satisfy the regularity conditions of MLEs. However, in this case, the possible value of the shape parameter is out of this range. Considering this, I limit the value of the shape parameter within (-1, 1) instead of (-0.5, 0.5) to capture more possible values. As shown in Figures 6.6 to 6.10, my study is similar to the results of Huerta and Stark (2012).

Figure 6.6 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the observations of the maximum monthly rainfall values data. For all the three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line is the posterior median values. The solid line in blue color stands for the posterior median values of the particle filter approximation and the dashed lines in blue color cover the 95% credible interval for the particle filter methods. Evidently, all three particle filters achieve similar results compared to the MCMC method. Furthermore, it can be seen that the estimated values from particle filter methods are larger than the estimates from the MCMC methods. Figure 6.7 shows estimates of the posterior distribution for the scale parameter $p(\sigma|y_{1:t}), t = 11, ..., T$. For all three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line are the posterior median values. The solid line in blue stands for the posterior median values of different particle filter approximation and the dashed lines in blue provide the 95% credible interval for the particle filter methods. All the estimates from particle filters show similar behavior as the MCMC method but the estimates are larger compared to the MCMC method. The approximation of the LW particle filter in plot (c) provides a stable estimation with a smaller length for the 95% credible interval. The approximation from the MCMC method provides information that the samples of the posterior of the scale parameter span between 7 to 10 with median values at around 9. This result agrees with the one obtained by Huerta and Stark (2012). The approximation from the LW particle filter provides samples at larger value and these samples span between 10 to 12 with the median values at around 11. Figure 6.8 shows estimates of the posterior distribution for the shape parameter $p(\xi|y_{1:t}), t = 11, ..., T$. In all three plots, the gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue stands for the posterior median values of

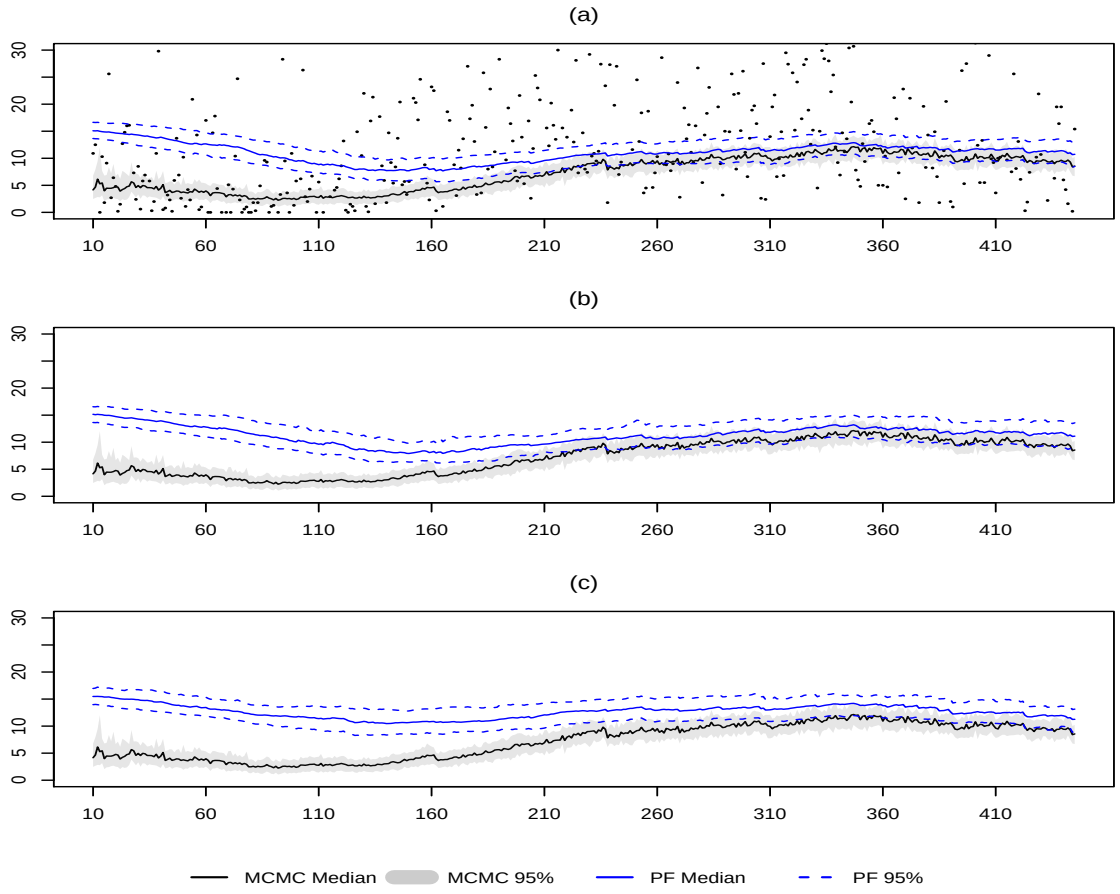

Figure 6.6: Estimation of the location parameter μ_t of the maximum monthly rainfall values using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

the BS particle filter approximation and the dashed lines in blue provide the 95% credible interval for the particle filter methods. In all these plots, the 95% credible intervals of the shape parameter do not contain zero. The posterior median values and the 95% credible intervals of the particle filters tend to be consistent with the Estimation of thethe MCMC method for large number of observations ($t > 300$). Since the estimates from the LW particle filter are more stable compared to other methods, including the MCMC and the BS and APF particle filter methods, we

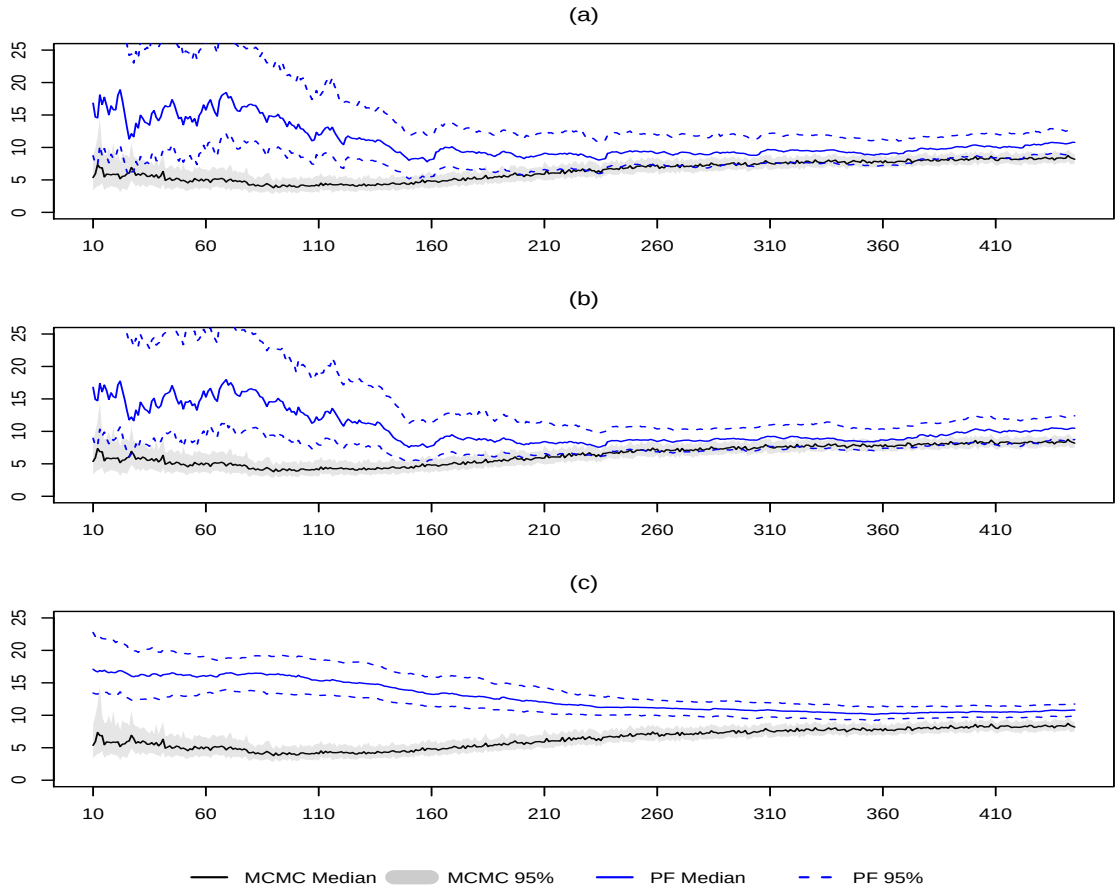

Figure 6.7: Estimation of the scale parameter σ of the maximum monthly rainfall values using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

can consider the estimates from the LW particle filter are more effective in this case. Figure 6.9 illustrates some diagnostics of the methods based on the simulation. Plot (a) shows the M-H acceptance rate of the MCMC method for both the scale (in blue) and shape (in green) parameters. Both of them have an acceptance rate within the range of 40% to 60% for most of the time points. Plot (b) at the bottom shows the number of effective sample size from the three particle filter methods, with BS particle filter in blue, APF particle filter in green and LW

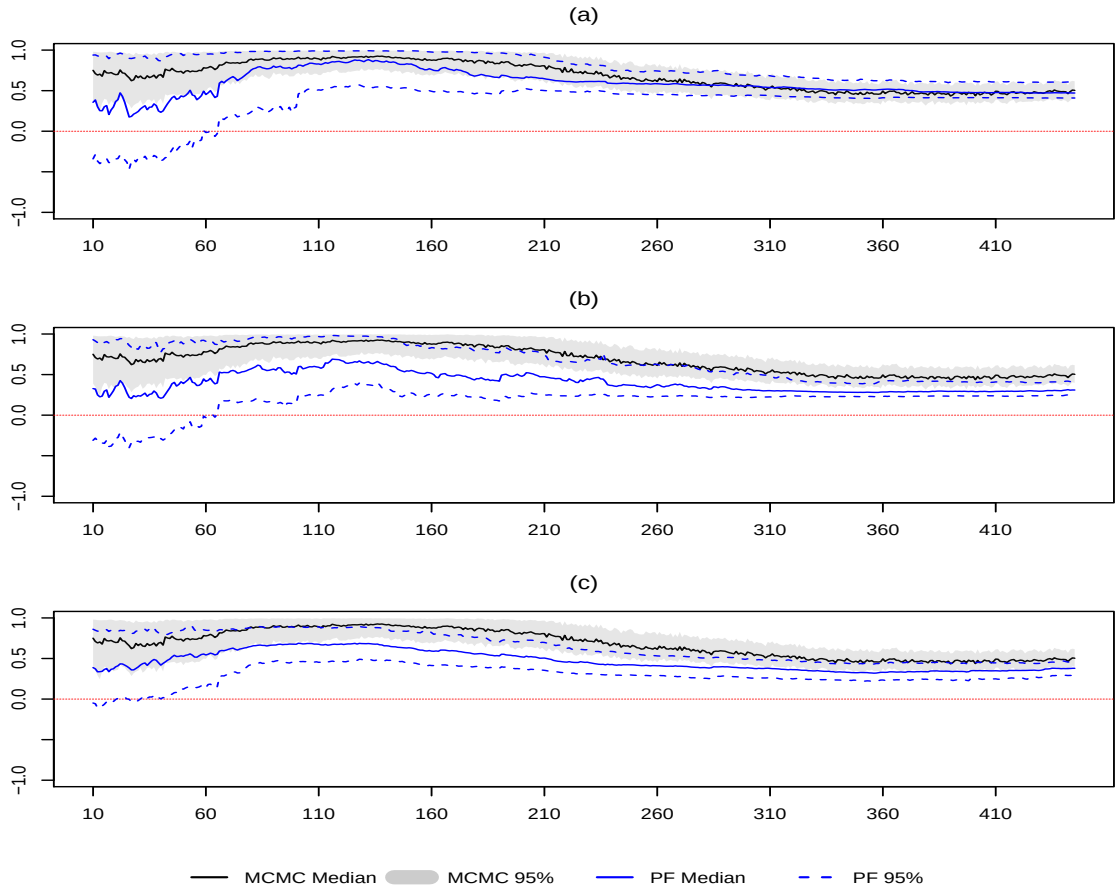

Figure 6.8: Estimation of the shape parameter ξ of the maximum monthly rainfall values using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

particle filter in purple respectively. In this plot, there are only a very limited number of sporadic drops in the effective sample size, but they return to normal quickly afterwards, suggesting that weight degeneracy is minor in this study. For most of the time points, the number of effective sample sizes from the APF and LW particle filter methods are close to the number of particles, 5000. Looking at the observations in plot (a) of Figure 6.6, all the observations are spread around all the space and no obvious outlier can be identified. Figure 6.10 shows histograms of

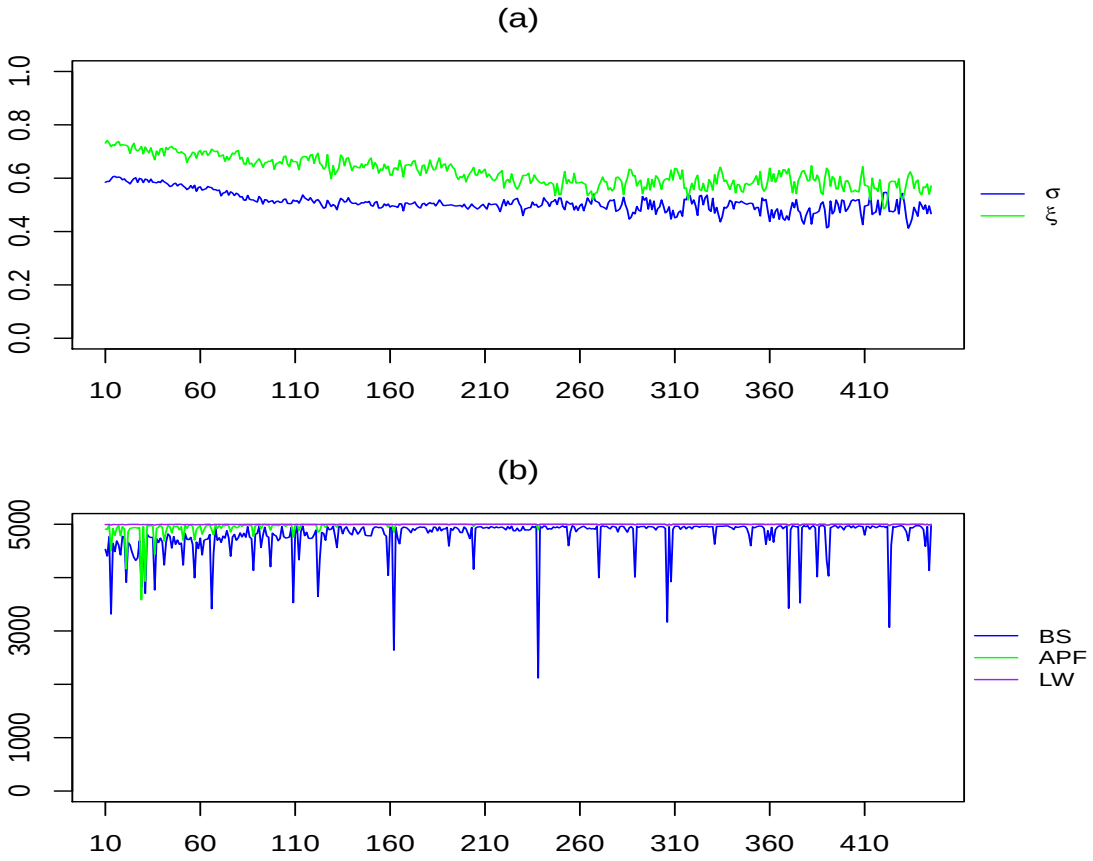

Figure 6.9: Diagnostics of the estimation of the maximum monthly rainfall values using a dynamic GEV model. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

samples from the marginal posterior approximation taken at the last time point, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ for all methods respectively. Histograms are limited to the same range in order to provide a clear comparison among the different methods. Histograms of the samples for $p(\mu_T|y_{1:T})$, in the left panel show all particle filters provide similar range and median values. The MCMC approximation presents smaller values in general compared to the particle filters. At the center panel, the histograms of the posterior distribution of the scale parameter

$p(\sigma|y_{1:T})$ show that the posterior median values of the three particle filters are consistent with each other while the LW particle filter has a smaller length for the 95% credible interval. The approximation of the MCMC method presents smaller values in general compared to the three particle filters. In the right panel, the histograms of the posterior distribution of the shape parameter $p(\xi|y_{1:T})$ show the posterior median values and the 95% credible intervals of all the methods are consistent with each other. Furthermore, none of the 95% credible intervals contain zero. The positive values demonstrate that under my modeling framework, these maximum monthly rainfall observations belong to a Frèchet distribution.

6.3 Minimum Daily Stock Returns

In this section, we apply our methods to the minimum daily stock returns occurring during a month using the Tokyo Stock Price Index (TOPIX). The original sample period is from January 4, 1990 to December 28, 2007. Currently, increasing numbers of researchers and practitioners are interested in high-frequency financial data to analyse the market dynamic structure. The daily stock return is one of the most popular figures that market participants much care about range of applications such as risk management and portfolio selection.

Nakajima, Kunihama, Omori and Frúhwirth-Schnatter (2012) studied this data with different GEV model that has state parameters with AR or MA evolution and they estimated the value of the scale parameter between 0.5 to 0.9. The estimated shape is also positive and less than 0.3. As mentioned in Nakajima, Kunihama, Omori and Frúhwirth-Schnatter (2012), the estimates of the parameter ξ obtained above imply that the underlying daily return data would follow a heavy-tailed distribution, as often pointed out in the financial literature. For $\xi > 0$, the GEV distribution gives the Frèchet distribution and its domain of attraction

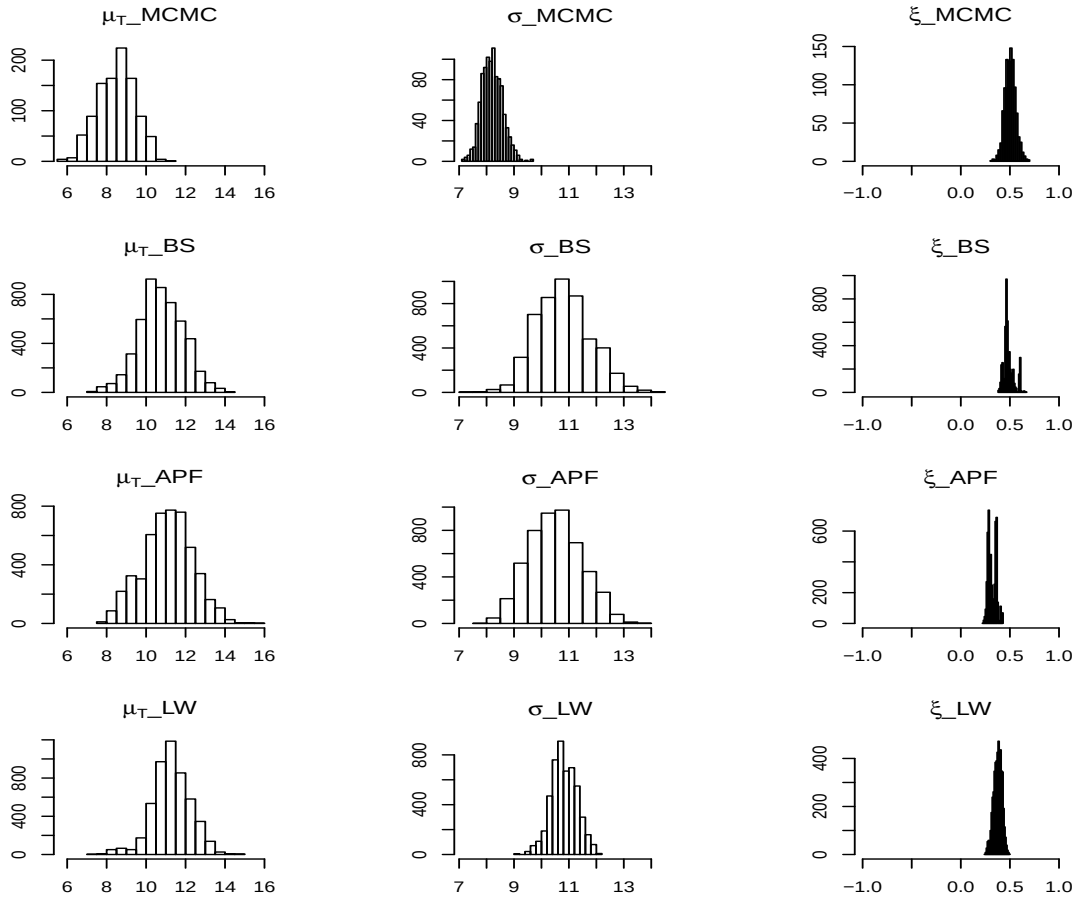

Figure 6.10: Histograms of posterior samples of the estimation of the maximum monthly rainfall values using a dynamic GEV model. (Rows are by method, columns are by parameter.)

includes distributions such as the Student-t, the Pareto and the Inverse-Gamma distributions. My results are similar to their results as shown in Figures 6.11 to 6.15.

Figure 6.11 shows the approximation of the posterior density of the location parameters $p(\mu_t|y_{1:t}), t = 11, ..., T$. In plot (a), the dots represent the observations from the minimum daily stock returns data. For all the three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid

black line provides the posterior median values. The solid line in blue stands for the posterior median values of the particle filter approximation and the dashed lines in blue are the 95% credible interval. The posterior median values and the 95% credible intervals from these particle filters have a close match to the MCMC method and small differences are hard to detect. Figure 6.12 shows estimates of

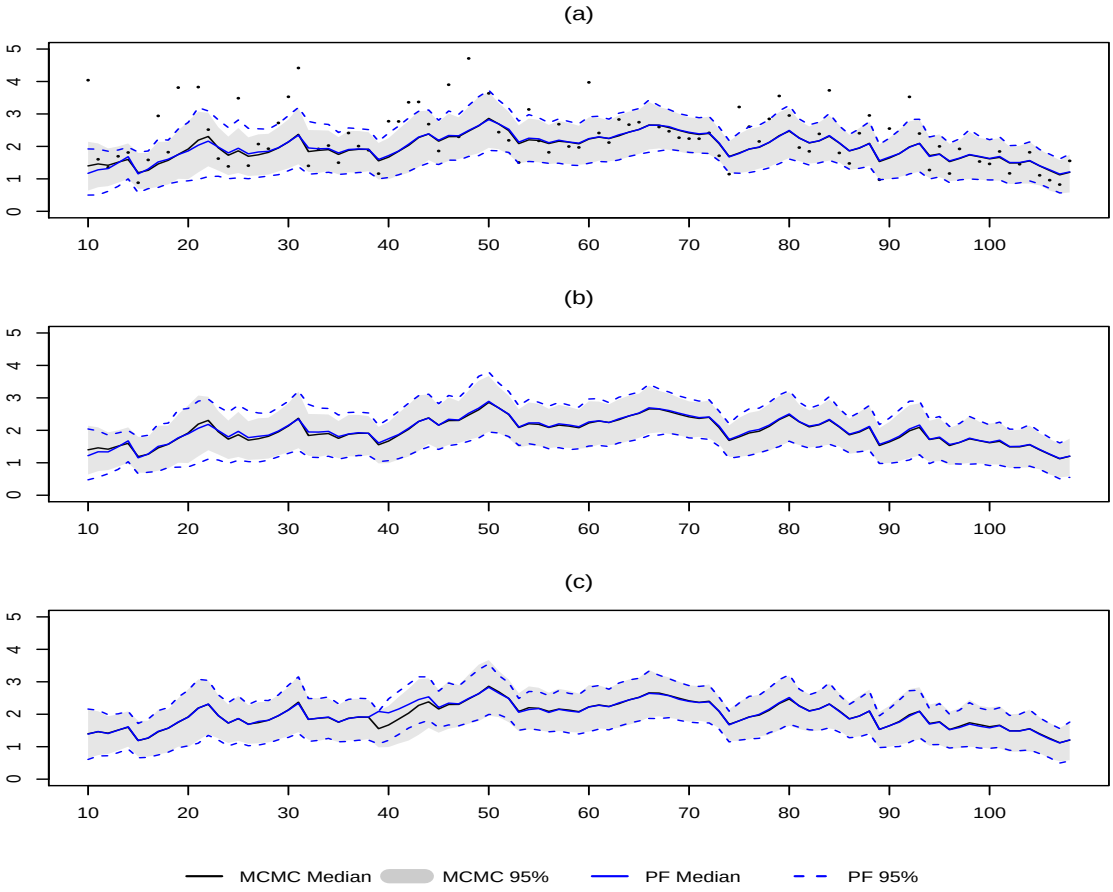

Figure 6.11: Estimation of the location parameter μ_t of the minimum daily stock returns, TOPIX data set, using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

the posterior distribution for the scale parameter $p(\sigma|y_{1:t}), t = 11, ..., T$. For all three plots, the gray area symbolizes the 95% credible interval of the MCMC sim-

ulation and the solid black line represents the posterior median values. The solid line in blue stands for the posterior median values of different particle filter approximation and the dashed lines in blue color cover the 95% credible interval for the particle filter methods. Overall, there is a close match between the particle filters and the MCMC method. It is demonstrated by these plots that the samples of the posterior approximation for the scale parameter span between 0.5 to 0.9 with the posterior median value being around 0.7. This result agrees with the numerical results obtained by Nakajima, Kunihama, Omori and Frúhwirth-Schnatter (2012). Figure 6.13 shows estimates of the posterior distribution for the shape parameter $p(\xi|y_{1:t}), t = 11, ..., T$. In all three plots, the filled gray area symbolizes the 95% credible interval of the MCMC simulation and the solid black line stands for the posterior median values. The solid line in blue represents the posterior median values of the BS particle filter approximation and the dashed lines in blue provide the 95% credible interval for the particle filter methods. Although these intervals contain zero for most of the time points, they contain positive value towards the end of the time series ($t > 100$). This result is similar to what I obtained in the simulation study of Section 5.3, where the samples from the shape parameter posterior approximation contain positive value only after a certain number of observations. Figure 6.14 illustrates some diagnostics of the methods based on the simulation. Plot (a) shows the M-H acceptance rate of the MCMC method for both the scale (in blue color) and shape (in green color) parameters. The scale parameter σ has acceptance rate within the range of 40% to 60%, and the shape parameter ξ has an acceptance rate that is slightly larger than 60%. Plot (b) at the bottom shows the effective sample size from the three particle filter methods, with BS particle filter in blue, APF particle filter in green and the LW particle filter in purple, respectively. In this plot, there are sporadic drops in the effective sample size, (see plot (a) of Figure 6.11), but they return to more acceptable values quickly afterwards, suggesting that weight degeneracy is not relevant in

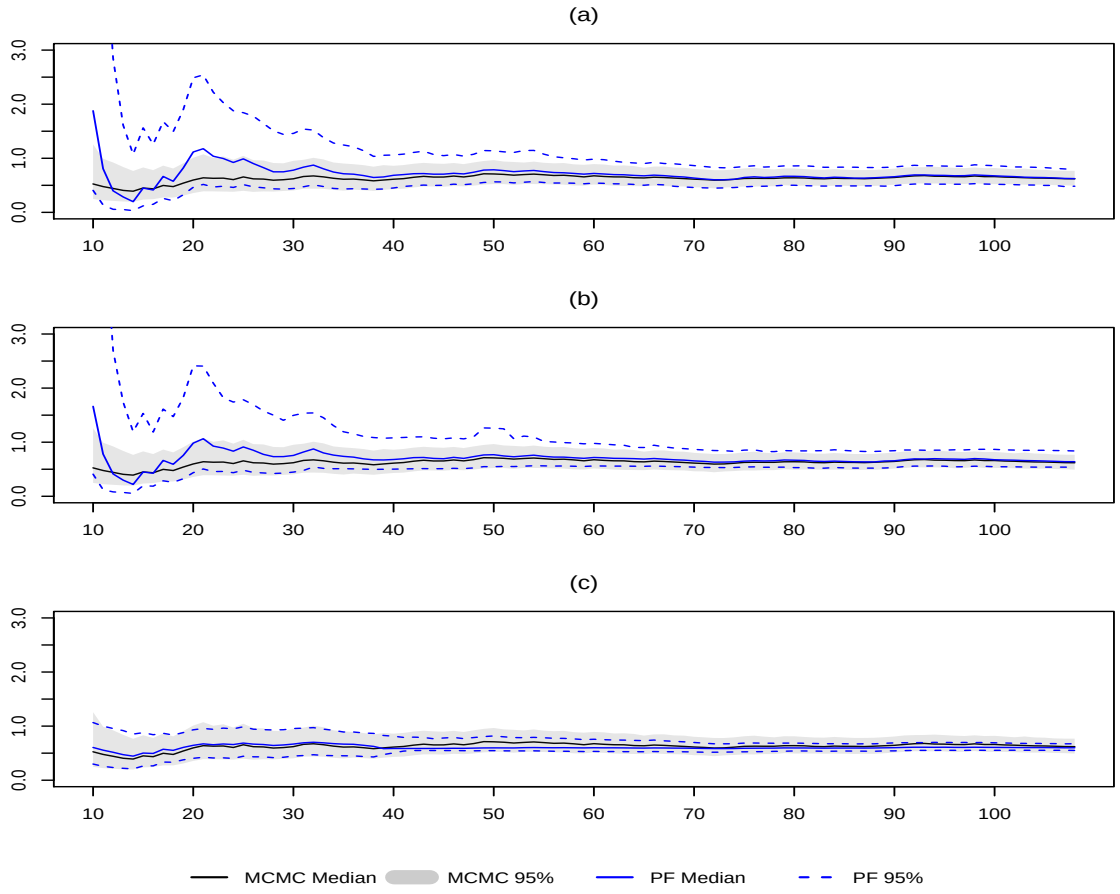

Figure 6.12: Estimation of the scale parameter σ of the minimum daily stock returns, TOPIX data set, using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

this study. Figure 6.15 shows histograms of samples from the marginal posterior approximation taken at the last time point, $p(\mu_T|y_{1:T})$, $p(\sigma|y_{1:T})$ and $p(\xi|y_{1:T})$ of all the methods respectively. Histograms are limited to the same range in order to provide a clear comparison among the different methods. Histograms of the samples for $p(\mu_T|y_{1:T})$, in the left panel show all methods provide similar range and median values. At the center panel, the histograms of samples of the posterior distribution of the scale parameter $p(\sigma|y_{1:T})$ also provide similar range and

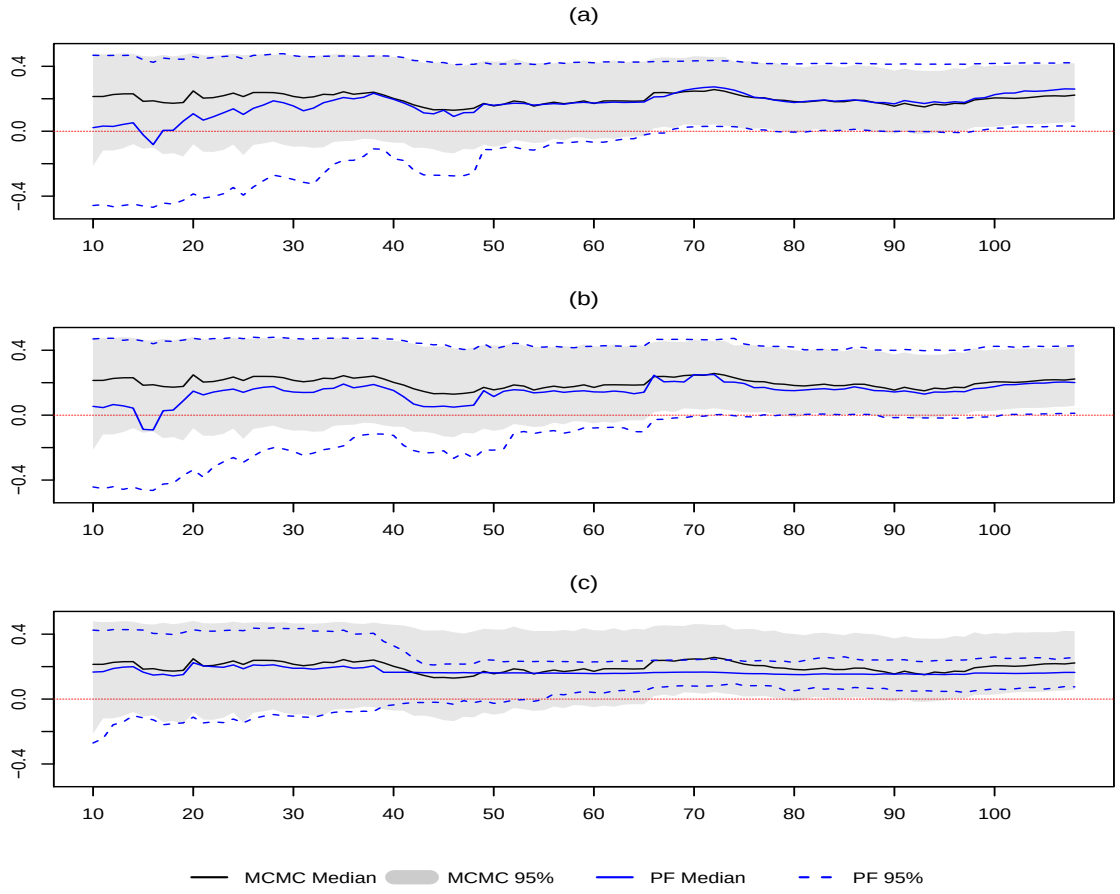

Figure 6.13: Estimation of the shape parameter ξ of the minimum daily stock returns, TOPIX data set, using a dynamic GEV model. (a) MCMC vs BS, (b) MCMC vs APF, (c) MCMC vs LW.

median values across all methods. In the right panel, the histograms of samples of the posterior distribution of the shape parameter $p(\xi|y_{1:T})$ show all the 95% credible intervals contain zero, but tend to contain majorly positive values.

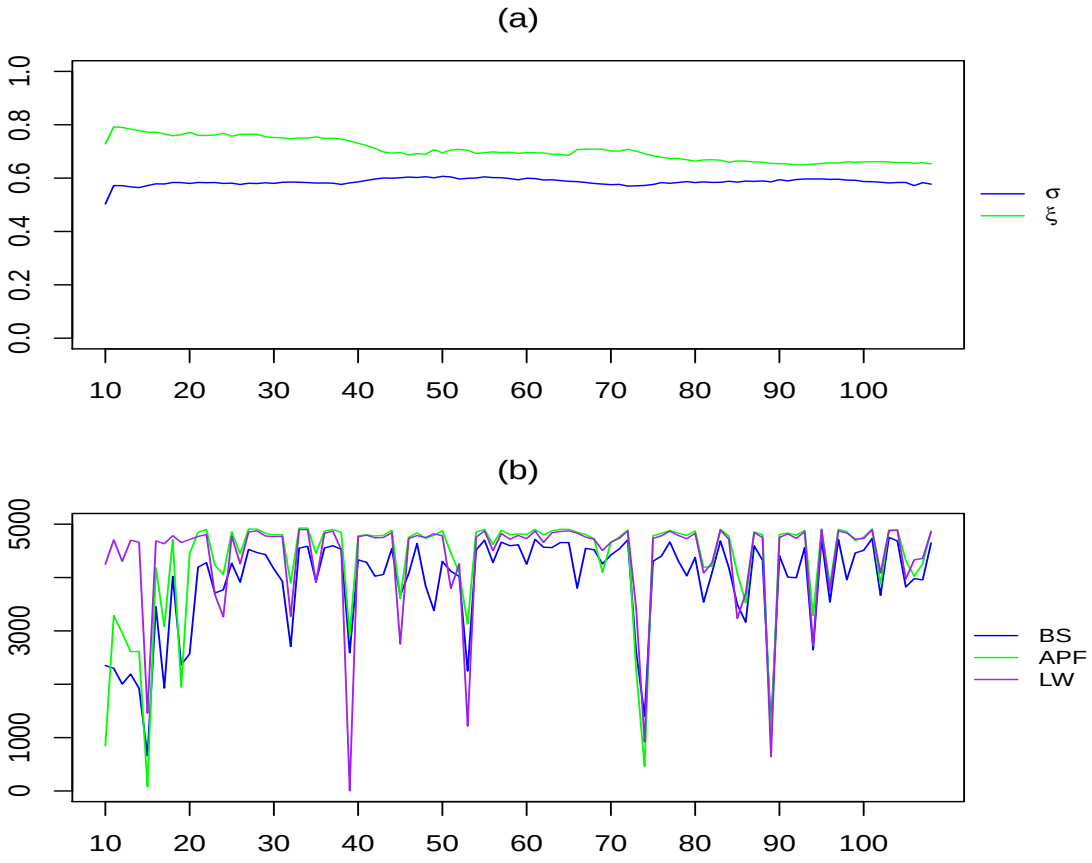

Figure 6.14: Diagnostics for the estimation of the minimum daily stock returns, TOPIX data set, using a dynamic GEV model. (a) MCMC M-H acceptance rate, (b) P.F. effective sample size.

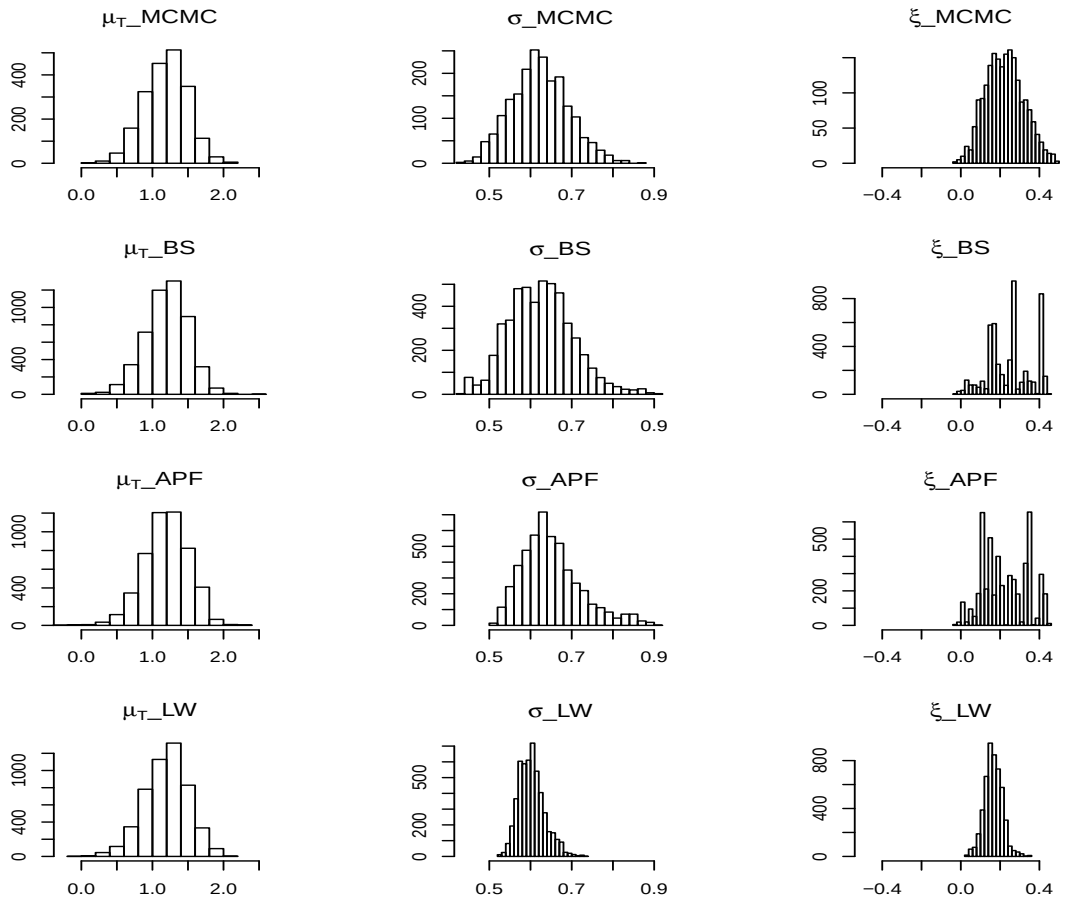

Figure 6.15: Histograms of posterior samples of minimum daily stock returns, TOPIX data set, using a dynamic GEV model. (Rows are by method, columns are by parameter.)

Chapter 7

Summary and Future Work

The aim of the thesis I submitted is to examine the performance of particle filters on a dynamic GEV model. There are papers talked about particle filter methodologies and applications on different data sets. A few papers discussed how to utilize particle filters to approximate the GEV model but there is lack of systematic comparison among all these algorithms. In this thesis, I first studied all the particle filters that are widely used and then introduced some recent development. A simple simulation study on the first order Dynamic Linear model showed that these particle filter algorithms could effectively estimate the parameters and in some cases even outperformed the MCMC method.

In Chapter 4 and 5, the dynamic GEV model was introduced and the posterior distribution was approximated applying these particle filter methodologies with comparison to the result of MCMC approximation. PCS density was proposed in the MCMC algorithm which largely improved the efficiency and decreased the computation time. The simulation study showed that our MCMC algorithm effectively approximated the location and scale parameters but had some issue to estimate the shape parameter.

We proposed the associated particle filter methods and generated particles directly

based on the data information and demonstrated that our filtering method was superior to the MCMC method in some cases. The application study in Chapter 6 illustrated several data sets in the time-dependent GEV class. The parameter estimations showed that applying particle filters on the time-dependent GEV models provided good fit to the data sets.

With respect to the future work, in addition to the marginal likelihood, forecasting performance is an important criterion for model comparison. This can be achieved by studying one step predictive density described as:

$$\tilde{F}(y) = \int F(y|\theta_{T+1}, x_{T+1}, x_{1:T}) p(\theta_{T+1}, x_{T+1}|y_{1:T}) d\theta_{T+1} dx_{T+1}. \tag{7.1}$$

Another future project will be on other proposed dynamic models, such as dynamic scale or shape parameters instead of making both fixed. Furthermore, some other model can be applied to the location parameters, such as a MA, RA or RW2 models. We'll need to compare which proposed model performs better. Discussed by Fulop and Li (2013), a sequential Bayes factor can be constructed for sequential model comparison, a feature that batch estimation cannot have. For any models M1 and M2, the Bayes factor at time t has the following recursive formula

$$\mathcal{BF}_t = \frac{p(y_{1:t}|M1)}{p(y_{1:t}|M2} = \frac{p(y_t|y_{1:t-1}, M1)}{p(y_t|y_{1:t-1}, M2)} \mathcal{BF}_{t-1}. \tag{7.2}$$

We can sequentially estimate $p(y_t|y_{1:t-1}, M)$ via

$$p(y_t|y_{1:t-1}, M) \approx \sum_{j=1}^{N} \omega_{t-1}^j p(y_t|x_{t-1}^j, \theta_{t-1}^j). \tag{7.3}$$

Another future topic will be studying the GEV model using the recently developed Integrated Nested Laplace Approximation (INLA) approach. When the data set is fixed, in the sense that all observations were already measured, and the interest is in the estimation of parameters and states using just this information, there is no reason why the procedure for inference should also be "dynamic". This is the idea pursued in the INLA approach, where the posteriors of interest are directly approximated avoiding look at the temporal structure of the data. INLA

is a computational approach recently introduced by Rue and Martino (2007) and Rue et al. (2009), to perform fast Bayesian inference in the broad class of latent Gaussian models. The INLA method does not sample from the posterior. It approximates the posterior with a closed form expression. Therefore, problems of convergence and mixing are not an issue. As an alternative to the usually time consuming MCMC methods, the main benefit of INLA approximations is computational: where Markov chain Monte Carlo algorithms need hours or days to run, INLA approximations provide more precise estimates in seconds or minutes. The GEV model package in INLA is recently developed and we will compare the performance between INLA and the particle filters as well.

Appendices

Appendix A

Detailed SIR Algorithm

Recall from section 2.1.1 that the one-step prediction distribution is given by

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{A.1}$$

The filtering distribution is

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \tag{A.2}$$

Proof:

$$\begin{aligned}
p(x_t|y_{1:t}) &= \frac{p(y_{1:t}, x_t)}{p(y_{1:t})} \\
&= \frac{p(y_{1:t}|x_t)p(x_t)}{p(y_{1:t})} \\
&= \frac{p(y_{1:t}|x_t)p(x_t)}{p(y_{1:t})} \\
&= \frac{p(y_t|x_t)p(y_{1:t-1}|x_t)p(x_t)}{\int p(y_{1:t}|x_t)p(x_t)dx_t} \\
&= \frac{p(y_t|x_t)p(y_{1:t-1}, x_t)/p(y_{1:t-1})}{\int p(y_{1:t}|x_t)p(x_t)dx_t/p(y_{1:t-1})} \\
&= \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(y_{1:t-1}|x_t)p(x_t)/p(y_{1:t-1})dx_t}
\end{aligned}$$

$$= \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}$$

The smoothing distribution:

$$p(x_t|y_{1:T}) = p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1} \tag{A.3}$$

Proof (Sarkka 2012)

Due to the Markov properties the state x_t is independent of $y_{t+1:T}$ given x_{t+1}, which gives $p(x_t|x_{t+1}, y_{1:T}) = p(x_t|x_{t+1}, y_{1:t})$.

$$\begin{aligned}
p(x_t|y_{1:T}) &= \int p(x_t, x_{t+1}|y_{1:T})dx_{t+1} \\
&= \int p(x_{t+1}|y_{1:T})p(x_t|x_{t+1}, y_{1:\mathbf{T}})dx_{t+1} \\
&= \int p(x_{t+1}|y_{1:T})p(x_t|x_{t+1}, y_{1:\mathbf{t}})dx_{t+1} \\
&= \int p(x_{t+1}|y_{1:T}) \frac{p(x_t, x_{t+1}|y_{1:t})}{p(x_{t+1}|y_{1:t})} dx_{t+1} \\
&= \int p(x_{t+1}|y_{1:T}) \frac{p(x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t})}{p(x_{t+1}|y_{1:t})} dx_{t+1} \\
&= \int p(x_{t+1}|y_{1:T}) \frac{p(x_{t+1}|x_t)p(x_t|y_{1:t})}{p(x_{t+1}|y_{1:t})} dx_{t+1} \\
&= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1} \\
&= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1}
\end{aligned}$$

A.1 Predictive Distribution

Particle estimation of the prediction density is based on the following approximation of relation A.1:

$$p(x_t|y_{1:t-1}) \simeq \frac{1}{N} \sum_{j=1}^{N} p_X(x_t|x_{j,t-1|t-1}) \tag{A.4}$$

In fact, on the grounds of this approximation we can obtain a random sample of $p(x_t|y_{1:t-1})$ by first choosing randomly an integer k in $\{1, ..., N\}$ and then generating a value from $p_X(x_t|x_{k,t-1|t-1})$.
Proof (Gordon et al. 1993):

- N is the total number of particles of $x_{t-1}^j, j = 1, ...N$ sampled from previous stage posterior density $p(x_{t-1}|y_{1:t-1})$, which will be the "prior" for current stage posterior density $p(x_t|y_{1:t})$.

- With N large enough, such as infinity. $\sum_{j=1}^{N} p(x_t|x_{t-1}^j)$ will be approximately equal to the total number of samples of x_t given all the possible x_{t-1}^j sampled from $p(x_{t-1}|y_{1:t-1})$. Then from the conditional distribution theory, $\frac{1}{N}\sum_{k=1}^{N} p(x_t|x_{t-1}^j)$ will be approximately equal to $p(x_t|y_{1:t-1})$.

A.2 Filtering Distribution

To generate the samples for the filtering distribution. That is, how to generate particles which follow the filtering density $p(x_t|y_{1:t})$.
Smith (1992) introduced how to re-sample from $H(X) = f(X)/\int f(X)dX$:

1. Given $\{X^j, j = 1, 2, ..., N\}$ are samples from $g(X)$, a density function easy to sample or already obtained from previous stage. Then define the Bootstrap weight, or mass as $q^j = \omega^j / \sum_j \omega^j$ with $\omega^j = f(X^j)/g(X^j)$.

2. Draw $X^\star$ from the discrete distribution over $X^1, X^2, ..., X^N$, and placing mass q^j on X^j.

3. Then $X^\star \sim H(X^\star)$ as the sample size N increase.

Back to our case, we want to re-sample from the filtering distribution

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}$$

which we can consider it as the $H(X) = f(X)/\int f(X)dX$ density we want to sample.

Then we follow Smith's procedure as described above:

1. Suppose $x^j, j = 1, 2, ..., N$ are samples from the prediction density function $g = p(x_t|y_{1:t-1})$. Define

$$\begin{aligned} \omega^j &= f(x^j)/g(x^j) \\ &= p(y_t|x^j)p(x^j|y_{1:t-1})/p(x^j|y_{1:t-1}) \\ &= p(x^j|y_{1:t}) \end{aligned}$$

   then the bootstrap weight is

$$\begin{aligned} q^j &= \omega^j / \sum_i \omega^j \\ &= \frac{p(y_t|x^j)}{\sum_{j=1}^N p(y_t|x^j)} \end{aligned}$$

2. Draw $\{x^{k[1]}, x^{k[2]}, ..., x^{k^{(j)}}, ..., x^{k[N]}\}$ from $\{x^1, x^2, ..., x^j, ..., x^N\}$ with mass q_i.

3. Then

$$\begin{aligned} x^{k^{(j)}} &\sim H(X) = f(X)/\int f(X)dX \\ &= \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1}dx_t} \end{aligned}$$

   which is just the filtering distribution function.

The filtering procedure can continue until the last time stage, to get samples of $p(x_T|y_{1:T})$.

A.3 Smoothing Distribution

We are assuming that $\{\overleftarrow{x_{t+1|T}}^j, j = 1, 2, ..., N\}$, are available, which denote the random samples generated from the marginal smoothing density $P(x_{t+1}|y_{1:T})$. Then we need backward recursively generate random draws from $P(x_t|y_{1:T})$, the previous time point given those samples from $p(x_{t+1}|y_{1:T})$.
The smoothing distribution:

$$p(x_t|y_{1:T}) = p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1}$$

is approximated by:

$$\begin{aligned}
p(x_t|y_{1:T}) &= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1} \\
&= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} p(x_{t+1}|y_{1:T}) dx_{t+1} \\
&\simeq p(x_t|y_{1:t}) \frac{1}{N} \sum_{j=1}^{N} \frac{p(\overleftarrow{x_{t+1|T}}^j|x_t)}{p(\overleftarrow{x_{t+1|T}}^j|y_{1:t})} \qquad \text{with } \overleftarrow{x_{t+1|T}}^j \sim p(x_{t+1}|y_{1:T}) \\
&\simeq p(x_t|y_{1:t}) \frac{1}{N} \sum_{j=1}^{N} \frac{p_x(\overleftarrow{x_{t+1|T}}^j|x_t)}{\frac{1}{N}\sum_{m=1}^{N} p(\overleftarrow{x_{t+1|T}}^m|x_t^m)} \qquad \text{with } x_t^m \sim p(x_t|y_{1:t}) \\
&= p(x_t|y_{1:t}) \sum_{j=1}^{N} \frac{p_x(\overleftarrow{x_{t+1|T}}^j|x_t)}{\sum_{m=1}^{N} p(\overleftarrow{x_{t+1|T}}^j|x_{m,t|t})}
\end{aligned}$$

A.4 PL Filtering State Full Conditional Density

$$\begin{aligned}
x_t &\sim p(x_t|s_{t-1}, x_{t-1}, V, W, y_t) \\
&= p(x_t|x_{t-1}, V, W, y_t) \\
&\propto p(y_t|x_t, V, W,)p(x_t|x_{t-1}, V, W,) \\
&\propto exp\left(-\frac{1}{2}\frac{(y_t - x_t)^2}{V} \right) \cdot exp\left(-\frac{1}{2}\frac{(x_t - x_{t-1})^2}{W} \right)
\end{aligned}$$

$$
\begin{aligned}
& \text{Because } y_t \sim N(x_t, V) \text{ and } x_t \sim N(x_{t-1}, W) \\
\propto \;& exp\left(-\frac{1}{2}\frac{W(y_t - x_t)^2 + V(x_t - x_{t-1})^2}{V \cdot W} \right) \\
= \;& exp\left(-\frac{1}{2}\frac{Wx_t^2 + Vx_t^2 - 2x_t y_t W - 2x_t x_{t-1} V + y_t^2 V + x_{t-1}^2 W}{V \cdot W} \right) \\
= \;& exp\left(-\frac{1}{2}\frac{(W+V)x_t^2 - 2x_t(y_t W + x_{t-1} V) + y_t^2 V + x_{t-1}^2 W}{V \cdot W} \right) \\
= \;& exp\left(-\frac{1}{2}\frac{x_t^2 - 2x_t \frac{(y_t W + x_{t-1} V)}{V+W} + \frac{y_t^2 V + x_{t-1}^2 W}{V+W}}{(V \cdot W)/(V+W)} \right) \\
\sim \;& N\left(\frac{(y_t W + x_{t-1} V)}{V+W}, \frac{V \cdot W}{V+W} \right)
\end{aligned}
$$

Appendix B

Programming Code

B.1 MCMC Code for Dynamic GEV

```
// Smooth function with all three parameters simulated
// provided the low 5% quantile as well
#include <cstdlib>
#include <iostream>
#include <algorithm>
#include <Rcpp.h>
#include <omp.h>
#include <vector>
#include <cmath>
//#include <random>
//#include "math.h"
//#include <pdflib.cpp>

using namespace std;
```

```
// Generate a random number between 0 and 1; uniform number
    in [0,1].
double unifRand() {
        return rand() / double(RAND_MAX);
 }

double stddev(std::vector<double> data, int n) {
        double mean = 0.0;
        double sum_deviation = 0.0;
        int i;
        for (i = 0; i<n; i++) {
                mean += data[i];
        }
        mean = mean / n;
        for (i = 0; i<n; i++)
                sum_deviation += (data[i] - mean)*(data[i]
                   - mean);
        return sqrt(sum_deviation / n);
}


double sampleNormal() {
        double u = ((double)rand() / (RAND_MAX)) * 2 - 1;
        double v = ((double)rand() / (RAND_MAX)) * 2 - 1;
        double r = u * u + v * v;
        if (r == 0 || r > 1) return sampleNormal();
        double c = sqrt(-2 * log(r) / r);
```

```
        return u * c;
}



double r8_normal_pdf(double av, double sd, double rval)  {
//****************************************************************************

//    R8_NORMAL_PDF evaluates the PDF of a normal
   distribution.
//    Input, double AV, the mean value.
//    Input, double SD, the standard deviation.  0.0 < SD.
//    Input, double RVAL, the point where the PDF is
   evaluated.
//    Output, double R8_NORMAL_PDF, the value of the PDF at
    RVAL.
  double pi = 3.141592653589793;
  double rtemp;
  double value;

  if ( sd <= 0.0 ) {
    std::cout << "\n";
    std::cout << "R8_NORMAL_PDF - Fatal error!\n";
    std::cout << "  Standard deviation must be positive.\n
       ";
    exit ( 1 );
  }
```

```
  rtemp = ( rval - av ) * ( rval - av ) * 0.5 / ( sd * sd )
     ;
  value = exp ( - rtemp ) / sd / sqrt ( 2.0 * pi );
  return value;
}

/*****************************************************************

double dGEV(double y, double mu, double sg, double xi) {
        double value;
        double t;

        if (xi == 0.0) {
                t = exp(-(y - mu) / sg);
            value = (1 / sg) * pow(t, (xi + 1)) * exp(-t);
          }
        else if (xi > 0) {
                if (y > (mu-sg/xi) ){
                t = pow( (1 + ((y - mu) / sg)*xi), (-1 / xi
                   ) );
            value = (1 / sg) * pow(t, (xi + 1)) * exp(-t);
                }
                else {
                        value = 0.0;
                }
        }
        else {
                if (y < (mu-sg/xi)){
```

```
                t = pow( (1 + ((y - mu) / sg)*xi), (-1 / xi
                    ) );
            value = (1 / sg) * pow(t, (xi + 1)) * exp(-t);
                }
                else {
                        value = 0.0;
                }
        }

        return value;
}



/*************************************************************************

//double gvlik(vector<double> y, vector<double> mu, double
    sg, double xi, int T) {
double gvlik(double y[], double mu[], double sg, double xi,
    int T) {
        double logliks = 0;
        for (int t = 0; t < T; t++) {
                logliks = logliks + log(fmin(pow(10, 300),
                    fmax(pow(10, -300), dGEV(y[t], mu[t], sg
                    , xi))));
        }
        return logliks;
}
```

```
/*****************************************************************

double gensg(double y[], double mu[], double sg, double xi,
    double ssig, int T) {
        double psin = log(sg) + ssig*sampleNormal();
        double sgex = exp(psin);

        int count = 0;
        while (sgex == 0 || sgex >30){
                psin = log(sg) + ssig*sampleNormal();
                sgex = exp(psin);
                count++;
                if (count > 10000){
                        sgex = 10 * unifRand();
                }
        }

        double priornew = log(fmax(pow(10, -300),
           r8_normal_pdf(0, 100, log(sgex))));
        double priorold = log(fmax(pow(10, -300),
           r8_normal_pdf(0, 100, log(sg))));

        double liknew = gvlik(y, mu, sgex, xi, T);
        double likold = gvlik(y, mu, sg, xi, T);

        double logratio = liknew - likold + priornew -
           priorold;
```

```
        if (_isnan(logratio) != 0 || (_finite(logratio)) ==
            0) {
                sg = sgex;
        }
        else {
                double un = log(unifRand());
                if (un <= fmin(logratio, 0)) {
                        sg = sgex;
                }
                else {
                        sg = sg;
                }
        }
return sg;
}

/*****************************************************************

double genxi(double y[], double mu[], double sg, double xi,
    double sxi, int T) {
        double randomNumber = sampleNormal();
        double xin = xi + sxi*randomNumber;


        int count = 0;
        while (xin < -0.5 || xin >0.5)  {
                xin = xi + sxi*sampleNormal();
                count++;
```

```
            if (count > 10000){
                    xin = unifRand() - 0.5;
            }

    }

    double h_star_t = fmax(pow(10, -300), r8_normal_pdf
       (xi, sxi, xin));
    double h_t_star = fmax(pow(10, -300), r8_normal_pdf
       (xin, sxi, xi));

    double priornew = log(max(pow(10, -300),
       r8_normal_pdf(0, 1000, xin)));
    double priorold = log(max(pow(10, -300),
       r8_normal_pdf(0, 1000, xi)));

    double liknew = gvlik(y, mu, sg, xin, T);
    double likold = gvlik(y, mu, sg, xi, T);
    double logratio = liknew - likold + priornew -
       priorold;

    double ratio = exp(logratio) * h_t_star / h_star_t;

    if (_isnan(ratio) != 0 || (_finite(ratio)) == 0) {
            xi = xin;
    }
    else {
            double un = unifRand();
```

```
                if (un <= fmin(ratio, 1)) {
                        xi = xin;
                }
                else {
                        xi = xi;
                }
        }
        return xi;
}
/*****************************************************************

//double mulik(vector<double> y, vector<double> mu, double
   sdmu, int T) {
double mulik(double y[], double mu[], double sdmu, int T) {
  double logliks = 0;
  for (int t = 1; t < T; t++) {
          logliks = logliks + log(fmin(pow(10, 300), max(
             pow(10, -300), r8_normal_pdf(mu[t - 1], sdmu,
             mu[t]))));
      }
  return logliks;
}

/*****************************************************************

void genmu(double y[], double * mu0_add, double sg, double
   xi, double sdmu, int T) {
        double munew, log_ratio;
```

```
double * mu[T];

for (int i = 0; i < T; i++)     {
        mu[i] = mu0_add++; /* assign the address of
            integer. */
}

// In the middle, it depends on obs before and
   after it.
for (int t = 1; t < T-1; t++) {
        munew = 1.0/2.0*(*mu[t-1] + *mu[t+1]) +
           sdmu/sqrt(2) * sampleNormal();
        log_ratio = log(dGEV(y[t], munew, sg, xi))
           - log(dGEV(y[t], *mu[t], sg, xi)) ;

        if (_isnan(log_ratio) != 0 || (_finite(
           log_ratio)) == 0) {
                *mu[t] = munew;
        }
        else {
                double un = log(unifRand());
                if (un <= fmin(log_ratio, 0)) {
                        *mu[t] = munew;
                }
                else {
                        *mu[t] = *mu[t];
                }
        }
```

```
}

// t=0, rely on t=1 only
munew = *mu[1] + sdmu * sampleNormal();
log_ratio = log(dGEV(y[0], munew, sg, xi)) - log(
   dGEV(y[0], *mu[0], sg, xi)) ;

if (_isnan(log_ratio) != 0 || (_finite(log_ratio))
   == 0) {
        *mu[0] = munew;
}
else {
        double un = log(unifRand());
        if (un <= fmin(log_ratio, 0)) {
                *mu[0] = munew;
        }
        else {
                *mu[0] = *mu[0];
        }
}

// t=T-1, rely on t=T-2 only
munew = *mu[T-2] + sdmu * sampleNormal();
log_ratio = log(dGEV(y[T-1], munew, sg, xi)) - log(
   dGEV(y[T-1], *mu[T-1], sg, xi)) ;

if (_isnan(log_ratio) != 0 || (_finite(log_ratio))
   == 0) {
```

```
                *mu[T-1] = munew;
        }
        else {
                double un = log(unifRand());
                if (un <= fmin(log_ratio, 0)) {
                        *mu[T-1] = munew;
                }
                else {
                        *mu[T-1] = *mu[T-1];
                }
        }
        //return mu;
}

/*****************************************************************************

/*****************************************************************************

/*****************************************************************************

using namespace std;
// [[Rcpp::export]]
Rcpp::NumericVector  MCMCRW1_sdmu(Rcpp::NumericVector y,
   Rcpp::NumericVector ITEs, Rcpp::NumericVector Initials)
   {
    int iter = ITEs[0];
        int burnin = ITEs[1];
        int thin = ITEs[2];
```

```
        int nthin = ITEs[3];
        int T = ITEs[4];
        double mu_0 = Initials[0];
        double sdmu = Initials[1];
        double scale_0 = Initials[2];
        double ssig = Initials[3];
        double xi = Initials[4];
        double sxi = Initials[5];

        double yy[T];
        for (int k = 0; k<T; k++) {
                yy[k] = y[k];
        }

    double sg = scale_0;

    double mu[T];
    for (int k = 0; k<T; k++) {
            mu[k] = mu_0 + sdmu*sampleNormal();
    }

    double mus[nthin][T];
    memset(mus, 0, (nthin)*T*sizeof(double));  //Initial
       matrix
    double scales[nthin];
    memset(scales, 0, (nthin)*sizeof(double));  //Initial
       matrix
    double shapes[nthin];
```

```
    memset(shapes, 0, (nthin)*sizeof(double));  //Initial
       matrix

    std::vector<double> log_scale_vec;
    std::vector<double> shape_vec;
    log_scale_vec.assign(1, log(sg));
    shape_vec.assign(1, xi);

    int K_thin = 0;
    int K_Accept_scale = 0;
    int K_Accept_shape = 0;
    double sg_old;
    double xi_old;

 for (int i=0; i<iter; i++) {
          sg_old = sg;
          xi_old = xi;

          genmu(yy, &mu[0], sg, xi, sdmu, T);
          genmu(yy, &mu[0], sg, xi, sdmu, T);  // Repeat
             here to use the new approx again  4/2/2015
          sg = gensg(yy, mu, sg, xi, ssig, T);
          xi = genxi(yy, mu, sg, xi, sxi, T);


          if (sg_old != sg) {
                K_Accept_scale = K_Accept_scale + 1;
          }
```

```
if (xi_old != xi) {
        K_Accept_shape = K_Accept_shape + 1;
}

std::vector<double>::iterator it;
it = log_scale_vec.end();
log_scale_vec.insert(it, 1, log(sg));

it = shape_vec.end();
shape_vec.insert(it, 1, xi);

std::vector<double> V1, V2;
if (i > 1000){
        V1.insert(V1.begin(), log_scale_vec.end() -
            499, log_scale_vec.end());
        V2.insert(V2.begin(), shape_vec.end() -
           499, shape_vec.end());
        ssig = fmax(stddev(V1, 500), 0.001);
        sxi = fmax(stddev(V2, 500), 0.001);

        V1.clear();
        V2.clear();
}

if (i >= (burnin-1) && (i - burnin + 1) % thin ==
   0){
        for (int j = 0; j < T; j++){
                mus[K_thin][j] = mu[j];
```

```
                }
                scales[K_thin] = sg;
                shapes[K_thin] = xi;
                K_thin++;
                }
}

Rcpp::NumericMatrix product(nthin, T+3);
// #pragma omp parallel for

  for (int i=0; i<nthin; i++) {
          for (int j = 0; j < T; j++){
                  product(i,j) = mus[i][j];
          }
  }

for (int i=0; i<nthin; i++){
          product(i,T)    = scales[i];
          product(i, T+1) = shapes[i];
          }

product(0, T + 2) = double(K_Accept_scale) / double(iter);
product(1, T + 2) = double(K_Accept_shape) / double(iter);
for (int j = 2; j < nthin; j++){
        product(j, T + 2) = 0;
}

return(product);
```

```
}
```

B.2 P.F. Code for Dynamic GEV

B.2.1 BS Filter

```
######## BootStrap (SIR) smoothing
    ##########################################################################

BS = function(y, mu_0, sig_loc, scale_0, sig_log_sca_0,
    shape_0, sxi_0, N, RW){
   T_Local = length(y)

   mus    = matrix(0,N)
   scales = matrix(0,N)
   shapes = matrix(0,N)

   mus_F    = matrix(0,N,T_Local)
   scales_F = matrix(0,N,T_Local)
   shapes_F = matrix(0,N,T_Local)
   weight_F = matrix(0,N,T_Local)

   mus_S    = matrix(0,N,T_Local)
   scales_S = matrix(0,N,T_Local)
   shapes_S = matrix(0,N,T_Local)
   weight_S = matrix(0,N,T_Local)

  ### Initial parameter values
```

```
sig_log_sca = sig_log_sca_0
sxi = sxi_0

loc_s = rnorm(N, mean=mu_0, sd=sig_loc)
#sca_s = exp(rnorm(N, mean=log(scale_0), sd=sig_log_sca))
sca_s =exp(rtruncnorm(n=N, a=-10^(300), b=log(30), mean=
    log(scale_0), sd=sig_log_sca))
sha_s = rtruncnorm(n=N, a=-0.5, b=0.5, mean=shape_0, sd=
    sxi)

if (RW==1) {
  ## Instead, t=1 make RW around itself
  for (t in 1:1) {
    mus    = rnorm(N, loc_s, sig_loc)
    #scales = exp(rnorm(N, log(sca_s), sig_log_sca))
    ### Use same scale and shape limit as in MCMC
    scales = exp(rtruncnorm(n=N, a=-10^(300), b=log(30),
       mean=log(sca_s), sd=sig_log_sca))
    shapes = rtruncnorm(n=N, a=-10, b=10, mean=sha_s, sd=
       sxi)

    w = NULL
    ### Shape bound removed due to NA values
    for (i in 1:N) {
     w[i] <- min(10^(300),
                 max(10^(-300),
                     dgev(y[t], loc=mus[i], scale=scales[i
                        ], shape=shapes[i], log = FALSE)))
```

```
                                }

    w = w/sum(w)
    ## Resampling
    index <- sample(1:N, size=N, replace=TRUE, prob=w)

    ## Save result
    mus_F[, t]     = mus[index]
    scales_F[, t] = scales[index]
    shapes_F[, t] = shapes[index]
    weight_F[, t] = rep(1/N, N)
  }

  for (t in 2:T_Local ){
    sxi = max(sxi*0.9, 0.01)
    sig_log_sca = max(sig_log_sca *0.9, 0.01)

    #### Sample vectors at right side are from last time
       iteration point (t-1)
    mus     = rnorm(N, mus_F[, t-1], sig_loc)
    #scales = exp(rnorm(N, log(scales_F[, t-1]),
       sig_log_sca))
    scales = exp(rtruncnorm(n=N, a=-10^(300), b=log(30),
       mean=log(scales_F[, t-1]), sd=sig_log_sca))
    shapes = rtruncnorm(n=N, a=-0.5, b=0.5, mean=shapes_F
       [, t-1], sd=sxi)

    w = NULL
```

```
      ### Shape bound removed due to NA values
      for (i in 1:N) {
        w[i] <- min(10^(300),
                    max(10^(-300),
                        dgev(y[t], loc=mus[i], scale=scales
                           [i], shape=shapes[i], log =
                           FALSE) )) }

      w = w/sum(w)
      ## Resampling
      index <- sample(1:N, size=N, replace=TRUE, prob=w)

      ## Save result
      mus_F[, t]    = mus[index]
      scales_F[, t] = scales[index]
      shapes_F[, t] = shapes[index]
      weight_F[, t] = rep(1/N, N)
    }
  }

  else if (RW==2) {
    ## First time point t=1&2, since they cannot apply RW2.
    ## Instead, t=1 make RW around itself and t=2 make RW
       around value of t=1
    for (t in 1:2 ){
      mus    = rnorm(N, loc_s, sig_loc)
      #scales = exp(rnorm(N, log(sca_s), sig_log_sca))
```

```
scales = exp(rtruncnorm(n=N, a=-10^(300), b=log(30),
    mean=log(sca_s), sd=sig_log_sca))
shapes = rtruncnorm(n=N, a=-0.5, b=0.5, mean=sha_s,
    sd=sxi)

w = NULL
### Shape bound removed due to NA values
for (i in 1:N) {
  w[i] <- min(10^(300),
              max(10^(-300),
                  dgev(y[t], loc=mus[i], scale=scales
                      [i], shape=shapes[i], log =
                      FALSE) ))  }

w = w/sum(w)
## Resampling
index <- sample(1:N, size=N, replace=TRUE, prob=w)

## Save result
mus_F[, t]    = mus[index]
scales_F[, t] = scales[index]
shapes_F[, t] = shapes[index]
weight_F[, t] = rep(1/N, N)

## Save theta_t=1 and being used at t=2 back to next
    loop.
loc_s = mus[index]
sca_s = scales[index]
```

```
    sha_s = shapes[index]
  }


  for (t in 3:T_Local ){
    sxi = max(sxi*0.9, 0.001)
    sig_log_sca = max(sig_log_sca *0.9, 0.001)
    #### Sample vectors at right side are from last time
       iteration point (t-1)
    mus     = rnorm(N, 2*mus_F[,t-1] - mus_F[, t-2],
       sig_loc)
    #scales = exp(rnorm(N, log(scales_F[, t-1]),
       sig_log_sca))
    scales = exp(rtruncnorm(n=N, a=-10^(300), b=log(30),
       mean=log(scales_F[, t-1]), sd=sig_log_sca))
    shapes = rtruncnorm(n=N, a=-0.5, b=0.5, mean=shapes_F
       [,t-1], sd=sxi)

    w = NULL
    ### Shape bound removed due to NA values
    for (i in 1:N) {
      w[i] <- min(10^(300),
                  max(10^(-300),
                      dgev(y[t], loc=mus[i], scale=scales
                         [i], shape=shapes[i], log =
                         FALSE))) }

    w = w/sum(w)
```

```
        ## Resampling
        index <- sample(1:N, size=N, replace=TRUE, prob=w)

        ## Save result
        mus_F[, t]     = mus[index]
        scales_F[, t] = scales[index]
        shapes_F[, t] = shapes[index]
        weight_F[, t] = rep(1/N, N)
      }

    }

    print("BS end")
    return(list(mus=mus_F, scales = scales_F, shapes =
       shapes_F, sig_log_sca=sig_log_sca, sig_sha=sxi))
}
```

B.2.2 APF Filter

```
######## Prado West P170 ######
## Try RW1 of location, scale and shape fixed ##

APF = function(y, mu_0, sig_loc, scale_0, sig_log_sca_0,
    shape_0, sxi_0, N, RW){
   T_Local= length(y)

  ### Initial parameter values
   sig_log_sca = sig_log_sca_0
   sxi = sxi_0
```

```
loc_s = rnorm(N, mean=mu_0, sd=sig_loc)
#sca_s = exp(rnorm(N, mean=log(scale_0), sd=sig_log_sca))
sca_s = exp(rtruncnorm(n=N, a=-10^(300), b=log(30), mean=
    log(scale_0), sd=sig_log_sca))
sha_s = rtruncnorm(n=N, a=-0.5, b=0.5, mean=shape_0, sd=
    sxi)

### Save result for mu, each column for each time point
    ###
mus_F    = matrix(0,N,T_Local)
scales_F = matrix(0,N,T_Local)
shapes_F = matrix(0,N,T_Local)
weight_F = matrix(0,N,T_Local)

mus_S    = matrix(0,N,T_Local)
scales_S = matrix(0,N,T_Local)
shapes_S = matrix(0,N,T_Local)
weight_S = matrix(0,N,T_Local)

qs = NULL
w1 = rep(1/N, N) ## For initial weight
mus_F[,1] =  loc_s

for (t in 1:T_Local){
  sxi = max(sxi*0.9, 0.01)
  sig_log_sca = max(sig_log_sca *0.9, 0.01)

  # (1) sample Auxiliary variable/index
```

```
w = NULL
for (i in 1:N) {
      ### max here to make sure no 0 value
   w[i]  = min(10^(300),
               max(10^(-300), dgev(y[t], loc=loc_s[i],
                  scale=sca_s[i], shape=sha_s[i], log =
                   FALSE) * w1[i] )) }
w=w/sum(w)
k = sample(1:N, size=N, replace=TRUE, prob=w)

# (2) Sample new samples
if(RW==1){
  if (t==1) {
      loc_s1 = rnorm(N, mus_F[k, 1], sig_loc) }
  else {
      loc_s1 = rnorm(N, mus_F[k, t-1], sig_loc) }
}
else if (RW==2) {
  if (t==1 | t==2) {
    loc_s1 = rnorm(N, mus_F[k, 1], sig_loc) }
  else {
    loc_s1 = rnorm(N, 2*mus_F[k, t-1] - mus_F[k, t-2],
       sig_loc) }
}

#sca_s1 = exp(rnorm(N, log(sca_s[k]), sig_log_sca))
sca_s1 = exp(rtruncnorm(n=N, a=-10^(300), b=log(30),
   mean=log(sca_s[k]), sd=sig_log_sca))
```

```
sha_s1 = rtruncnorm(n=N, a=-0.5, b=0.5, mean=sha_s[k],
    sd=sxi)

# (3) Get new weights
w1 = NULL
for (i in 1:N) {
    w1[i]  = min(10^300,
                 max(10^(-300), dgev(y[t], loc=loc_s1[i
                     ], scale=sca_s1[i], shape=sha_s1[i],
                      log = FALSE) / w[k[i]] ))
  }
### w[k[i]] is the mean of ~ par_(t+1)|par_t[k], which
    is just par_t[k].



### Add extra step of resampling for each one
w1 = w1/sum(w1)
## Resampling
index <- sample(1:N, size=N, replace=TRUE, prob=w1)

## Save result
mus_F[, t]    = loc_s1[index]
scales_F[, t] = sca_s1[index]
shapes_F[, t] = sha_s1[index]
weight_F[, t] = rep(1/N, N)

loc_s = loc_s1[index]
sca_s = sca_s1[index]
```

```
    sha_s = sha_s1[index]
    w1 = rep(1/N, N)
  }

print("APF end")
  return(list(mus=mus_F, scales = scales_F, shapes =
    shapes_F, sig_log_sca=sig_log_sca, sig_sha=sxi))
}
```

B.2.3 LW Filter

```
LW = function(y, mu_0, sig_loc, scale_0, sig_log_sca_0,
   shape_0, sxi_0, N, RW){
  T_Local  = length(y)
  h2 = 1-a^2

  ### Initial parameter values
  sig_log_sca = sig_log_sca_0
  sxi = sxi_0

  loc_s = loc_s_0 = rnorm(N, mean=mu_0, sd=sig_loc)  # rep(
    mu_0, N)
  #sca_s = exp(rnorm(N, mean=log(scale_0), sd=sig_log_sca))
  sca_s = exp(rtruncnorm(n=N, a=-10^(300), b=log(30), mean=
    log(scale_0), sd=sig_log_sca))
  sha_s = rtruncnorm(n=N, a=-0.5, b=0.5, mean=shape_0, sd=
    sxi)
```

```
pars = cbind(log(sca_s), sha_s)   ## Below is the fixed
    parameters \phi
w = rep(1/N, N)

### Save result for smoothing useage, each column for
    each time point###
mus_F    = matrix(0,N,T_Local)
scales_F = matrix(0,N,T_Local)
shapes_F = matrix(0,N,T_Local)
weight_F = matrix(0,N,T_Local)

mus_S    = matrix(0,N,T_Local)
scales_S = matrix(0,N,T_Local)
shapes_S = matrix(0,N,T_Local)
weight_S = matrix(0,N,T_Local)

qs = NULL
for (t in 1:T_Local){
  vpar   = var(pars)
  # Step 1, identify mu_t and m_(t-1)
  ## Sample m of parameter vector:
  mpar  = apply(pars,2,mean)  ## Mean of each column,
     mean of phi(t-1)
  m      = a*pars+(1-a)*matrix(mpar,N,2,byrow=TRUE)    ##
     m_(t-1)
  m_log_sca_s  = m[,1]
  m_sha_s      = m[,2]
```

```
# Step 2, sample auxiliary index k
weight = NULL
for (i in 1:N) {
  weight[i]  = min(10^(300),
                   max(10^(-300),  dgev(y[t], loc=
                       loc_s_0[i], scale=exp(m_log_sca_s
                       [i]), shape=m_sha_s[i], log =
                       FALSE) ))
}

weight = weight/sum(weight)
k = sample(1:N, size=N, replace=TRUE, prob=weight)

# Step 3, sample fixed parameters: phi_t
Mat <- h2*vpar
newMat <- Mat + 0.000001*diag(2)
newMat <- 0.5*newMat + 0.5*t(newMat)

#pars = m[k,] + matrix(rnorm(2*N),N,2)%*%chol(newMat)
#pars = m[k,] + mvrnorm(n=N, mu=c(0,0), Sigma=h2*vpar,
    tol = 1e-6, empirical = FALSE)
pars = m[k,] + mvrnorm(n=N, mu=c(0,0), Sigma=newMat,
    tol=1e-6, empirical=FALSE)

log_sca_s  = pars[,1]
sha_s  = pars[,2]

## Step 4, sample state vector _{t+1}
```

```
if (RW==1) {
  if (t==1) {
    loc_s = rnorm(N, loc_s_0[k], sig_loc) }
  else {
    loc_s = rnorm(N, mus_F[k, t-1], sig_loc) }
}
else if (RW==2) {
  if (t==1 ){
    loc_s  = rnorm(N, loc_s_0[k], sig_loc) }
  else if (t==2) {
    loc_s  = rnorm(N, mus_F[k, t-1], sig_loc)  }
  else {
    loc_s  = rnorm(N, 2*mus_F[k, t-1] - mus_F[k, t-2],
       sig_loc) }
}
## Step 5
for (i in 1:N) {
  w[i]  = min(10^300,  max(10^(-300), dgev(y[t], loc=
     loc_s[i], scale=exp(log_sca_s[i]), shape=sha_s[i],
      log = FALSE)  /  weight[k[i]]
                                    * dunif(sha_s[i],
                                       -0.5, 0.5) ))  }
### Add extra step of resampling for each one

## Resampling
w <- w/sum(w)
index <- sample(1:N, size=N, replace=TRUE, prob=w)
```

```
    pars  = pars[index,]
    loc_s = loc_s[index]
    sca_s = exp(log_sca_s[index])
    sha_s = sha_s[index]

    mus_F[, t]    = loc_s
    scales_F[, t] = sca_s
    shapes_F[, t] = sha_s

    loc_s_0 = loc_s
  }

print("LW end")
return(list(mus=mus_F, scales = scales_F, shapes = shapes_F
   , pars = pars))
}
```

References

[1] Adlouni, E. S. and Ouarda, T., "Joint Bayesian model selection and parameter estimation the generalized extreme value model with covariates using birth-death Markov chain Monte Carlo", *Water Resources Research*, vol. 45, W06403, 2009.

[2] Andrieu, C., Doucet, A. and Holenstein, R., "Particle Markov chain Monte Carlo methods", *Journal of the Royal statistical Society Series B*, vol. 72, pp. 269-342, 2010.

[3] Ailliot, P., Thompson, C. and Thomson, P., "Mixed methods for fitting the GEV distribution", *Water Resources Research*, Vol.47, W05551, doi:10.1029/2010WR009417, 2011.

[4] Briers,M., Doucet, A. and Maskell, S., "Smoothing algorithms for state-space models", *Annals of the Institute of Statistical Mathematics*, vol. 62, Number 1, pp. 61-89, 2010.

[5] Cappe, O., Godsill, S.J. and Moulines, E., "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo", *Proceedings of the IEEE*, vol.95, iss 5, pp.899-924, 2007.

[6] Carlin, B., Polson, N. G. and Stoffer, D., "A Monte Carlo approach to nonnormal and nonlinear state-space modeling", *Journal of American Statistical Association*, vol.87, pp.493-500, 1992.

[7] Carter, C. and Kohn, R., "On Gibbs sampling for state space models", *Biometrika*, Vol.81, pp. 541-553, 1994.

[8] Carvalho, C. M., Johannes, M., Lopes, H. F. and Polson, N., "Particle learning and smoothing", *Statistical Science*, vol. 25, pp. 88-106, 2010.

[9] Coles, S., "An Introduction to Statistical Modeling of Extreme Values", *Springer Series in Statistics*, Springer-Verlag, 2001.

[10] Doucet, A., Freitas, N. and Gordon N., "Sequential Monte Carlo Methods in Practice", *Springer*, 2001.

[11] Doucet, A. and Johansen, A., "A Note on Auxiliary Particle Filters", *Statistics & Probability Letters*, vol. 78, pp.1498-1504, 2008.

[12] Doucet, A. and Johansen, A., "A Tutorial on Particle Filtering and Smoothing: Fifteen years later", *Oxford Handbook of Nonlinear Filtering*, 2008.

[13] Erol, Y., Li, Y., Ramsundar, B. and Russell, S., "The extended parameter filter", *arXiv preprint arXiv:1305.1704*, 2013.

[14] Fulopa, A. and Li, J., "Efficient learning via simulation: A marginalized resample-move approach", *Journal of Econometrics*, Vol 176, pp.146-161, 2013

[15] Gaetan, C. and Grigoletto, M., "Smoothing sample extremes with dynamic models", *Extremes*, 7, 221-236, 2004.

[16] Gelman, A., "Prior distributions for variance parameters in hierarchical models", *Bayesian Analysis*, 1, No.3, 515-533, 2006.

[17] Geweke, J. and Tanizaki, H., "On Markov Chain Monte Carlo Methods for Nonlinear and Non-Gaussian State-Space Models." *Communications in Statistics, Simulation and Computation*, Vol.28, No.4, pp. 867-894, 1999.

[18] Geweke, J. and Tanizaki, H., "Bayesian Estimation of State-Space Model Using the Metropolis-Hastings Algorithm within Gibbs Sampling." *Communications in Statistics, Simulation and Computation*, Vol.28, No.4, pp.867 - 894., 1999.

[19] Godsill, S.J., Doucet, A. and West, M., "Monte Carlo smoothing for nonlinear time series", *Journal of the American Statistical Association*, vol.50, pp.438 - 499, 2004.

[20] Gordon, N.J., Salmond, D.J. and Smith, A.F.M., "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", *IEE Proc.* F 140, 107-113, 1993.

[21] Huerta G. and Sansó B., "Time-Varying Models for Extreme Values", *Environmental and Ecological Statistics*. Vol. 14, No. 3, pp. 285-299, 2007.

[22] Huerta, G. and Stark, G.A., "Dynamic and spatial modeling of block maxima extremes", *Bayesian Inference and Markov Chain Monte Carlo: In Honor of Adrian Smith*, Oxford University Press, Ch. 10 pp. 183-199. 2012.

[23] Ignatious, J., UmaMageswari, A. and Lincon, A., "Adaptive Particle Filter Approach to Approximate Particle Degeneracy", *International Journal of Scientific & Engineering Research*, Volume 3, Issue 7, July-2012.

[24] Kantas, N., Doucet, A., Singh, S.S. and Maciejowski, J.M., "An Overview of Sequential Monte Carlo Methods for Parameter Estimation in General State-Space Models", *System Identification* Volume 15, Part 1, pp.774-785, 2009.

[25] Kitagawa, J., "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models", *Journal of Computational and Graphical Statistics*, Vol. 5, No. 1., pp. 1-25, 1996.

[26] Klaas, M., de Freitas. N. and Doucet. A., "Toward Practical N2 Monte Carlo: the Marginal Particle Filter", *Conference: Uncertainty in Artificial Intelligence*, pp. 308-315, 2005.

[27] Kotz, S. and Nadarajah, S., "Extreme Value Distributions: Theory and Applications", *Imperial College Press*, 2000.

[28] Lindsten, F., Schon, T.B. and Svensson, L., "A non-degenerate Rao-Blackwellised particle filter for estimating static parameters in dynamical models", *Proceedings of the 16th IFAC Symposium on System Identification*, Brussels, Belgium, 2012.

[29] Liu, J. S., "Metropolized independent sampling with comparisons to rejection sampling and importance sampling", *Statistics and Computing*, vol.6, iss 2, pp 113-119, 1996.

[30] Liu, J. S. and Chen. R., "Sequential Monte Carlo methods for dynamic systems", *Journal of the American Statistical Association*, Vol. 93, No. 443, pp. 1032-1044, 1998.

[31] Liu, J. and West, M., "Combined parameters and state estimation in simulation based filtering", *Sequential Monte Carlo Methods in Practice*, Springer-Verlag New York, pp. 197-223, 2001.

[32] Lopes, H. F., Carvalho, C. M., Johannes, M. and Polson, N. G., "Particle learning for sequential Bayesian computation (with discussion)", *Bayesian Statistics*, Oxford: Oxford University Press, pp.317-360, 2011.

[33] Lopes, H. F. and Tsay, R., "Particle filters and Bayesian inference in financial econometrics", *Journal of Forecasting*, vol. 30, pp. 168-209, 2011.

[34] Martins, T. G., Simpson, D., Lindgren, F. and Rue H., "Bayesian computing with INLA: new features", *Computational Statistics & Data Analysis*, Vol. 67, pp. 68-83, Nov-1999.

[35] Migon, H.S., Gamerman, D., Lopes, H. and Ferreira, M., "Dynamic models", *Handbook of Statistics, Vol. 25*, 2005.

[36] Nakajimaa,J., Kunihamaa, T., Omorib, Y. and Frhwirth-Schnatterc, S., "Generalized extreme value distribution with time-dependence using the AR and MA models in state space", *Computational Statistics & Data Analysis*, vol. 56, Issue 11, pp. 3241-3259, 2012.

[37] Nemeth, C., Fearnhead, P. and Mihaylova, L., "Particle approximations of the score and observed information matrix for parameter estimation in state space models with linear computational cost", arXiv:1306.0735, Cornell Uniersity Library, submitted paper, 2013.

[38] Petris, G., Petrone, S. and Campagnoli P., "Dynamic Linear Model with R", *Springer*, 2007.

[39] Pitt, M. and Shephard, N., "Filtering via Simulation: Auxiliary Particle Filters", *Journal of the American Statistical Association*, Vol. 94, No. 446. pp. 590-599, June-1999.

[40] Polson, N. G., Stroud, J. R. and Mullef, P., "Practical filtering with sequential parameter learning", *Journal of the Royal Statistical Society, series B*, 70, 413-28, 2008.

[41] Prado, R. and West, M., "Time Series - Modeling, Computation and Inference ", *Springer*, 2010.

[42] Robert, C. P., and Casella G., "Introducing Monte Carlo Methods with R", *CRC Press.*, 2009.

[43] Robinson, M. E. and Tawn, J. A., "Statistics for exceptional athletics records", *Appl. Stat.* vol.44, pp. 499-511, 1995.

[44] Robinson, M. E. and Tawn, J. A. "Letters to the Editor, Statistics for exceptional athletics records", *Applied Statistics Journal of the Royal Statistical Society C*, vol.46(1), pp. 127-128, 1997.

[45] Rue, H. and Martino, S., "Approximate Bayesian inference for hierarchical Gaussian Markov random field models", *Journal of statistical planning and inference*, Vol. 137(10), pp.3177-3192, 2007.

[46] Rue, H., Martino, S., and Chopin, N., "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71(2), pp.319-392, 2009.

[47] Ruiz-Cárdenas, R., Krainski, E., and Rue, H., "Direct fitting of dynamic models using integrated nested laplace approximations-inla", *Computational Statistics & Data Analysis*, 2011.

[48] Sarkka, S., "Bayesian filtering and smoothing", *Cambridge University Press*, 2013.

[49] Smith, R. L., "Maximum likelihood estimation in a class of nonregular cases", *Biometrika*, 72, 67-92, 1985.

[50] Smith, A.F.M. and Gelfand, A.E., "Bayesian Statistics without Tears: A Sampling-Resampling Perspective, " *The American Statistician*, Vol.46, No.2, pp.84 - 88, 1992.

[51] Smith, R. L. "Letters to the Editor, Statistics for exceptional athletics records", *Applied Statistics - Journal of the Royal Statistical Society C*, vol.46, No.1, pp. 123-127, 1997.

[52] Storvik, G., "Particle filters for state-space models with the presence of unknown static parameters", *IEEE Transactions on Signal Processing*, vol.50, pp. 281-289, 2002.

[53] Tanizaki, H., "Nonlinear and non-Gaussian state-space modeling with Monte Carlo techniques: A survey and comparative study," *Statistics (Stochastic Processes: Modeling and Simulation)* (C.R. Rao and D.N. Shanbhag, eds) North Holland, Amsterdam, 871-929, 2003.

[54] Taylor, B.M. and Diggle, P.J., "INLA or MCMC? A Tutorial and Comparative Evaluation for Spatial Prediction in log-Gaussian Cox Processes", *Journal of Statistical Computation and Simulation*, 2012.

[55] Whiteley, N. and Johansen, A., "Recent Developments in Auxiliary Particle Filtering", *Inference and Learning in Dynamic Models*, Cambridge University Press, 2011.