Data-Cloning SMC\textsuperscript{2} for Applications to Latent Variable Models

Jin-Chuan Duan\textsuperscript{*}, Andras Fulop\textsuperscript{†} and Yu-Wei Hsieh\textsuperscript{‡}

(First Draft: June 23, 2017; This Draft: June 1, 2018)

Abstract

Two applications are considered – (1) a mixed-logit model for decision making, and (2) a censored GARCH model for describing common stock returns in China where daily prices are subject to either 10 or 5% up/down price limits, depending on whether a common stock is under special treatment. A data-cloning SMC\textsuperscript{2} maximum likelihood estimation algorithm is proposed for these two applications, and also serves as a general-purpose optimization routine for models with latent variables. The idea is to first marginalize out latent variables by applying one layer of SMC at a fixed parameter value, and then estimate the model parameters by another layer of SMC utilizing density-tempering. Data-cloning is employed to effectively reduce Monte Carlo errors inherent in the SMC\textsuperscript{2} algorithm, and also to address multi-modality present in typical latent variable models. This new method has wide applicability and can be massively parallelized to take advantage of typical computers today. The two specific applications clearly demonstrate the power of this SMC\textsuperscript{2} algorithm.

Keywords: Sequential Monte Carlo; Data Clone; Latent Variable; Maximum Likelihood; Monte Carlo Optimization

JEL Classification: C15, C63

\textsuperscript{*}National University of Singapore (Business School, Risk Management Institute and Department of Economics). E-mail: bizdjc@nus.edu.sg

\textsuperscript{†}Department of Finance, ESSEC Business School. Email: fulop@essec.fr. Andras Fulop would like to thank the financial support of Labex MME-DII (ANR11-LBX-0023-01)

\textsuperscript{‡}Department of Economics and Dornsife Institute for New Economic Thinking, University of Southern California. Email: yuwei.hsieh@usc.edu. Yu-Wei Hsieh thanks Baiyu Dong and Xing Zhang for research assistance. An earlier version of this paper was released under the title of “Maximum Likelihood Estimation of Latent Variable Models by SMC with Marginalization and Data Cloning”.
1 Introduction

Two specific classes of latent variable models common in economics/finance are the focus of this paper. We will show that maximum likelihood estimation (MLE) for these two classes can be effectively handled with a new data-cloning SMC algorithm, and by extension many other models with latent variables can be likewise estimated. The first is a static mixed-logit model applied to the data on the fishing mode choices made by many decision makers. It is a stylized, yet computationally challenging example typical of the discrete-choice literature. The second is a dynamic GARCH model for stock returns subject to daily price limits such as in the Chinese stock markets where price limits significantly complicate volatility dynamics and thus parameter estimation.

Classical inference usually involves maximizing a likelihood function. Except for few cases, most modern statistical models lead to complicate nonlinear programming problems with possibly many local solutions. As a result, Monte Carlo optimization, such as Simulated Annealing (SA), is often adopted in practice. However, SA is inapplicable in latent variable models, and researchers often rely on two alternative approaches: (1) marginalizing out latent variables such as simulated maximum likelihood (SML), and (2) data augmentation by sampling $f(\theta, U)$ via MCMC, which is the joint distribution (posterior) of the parameters of interest $\theta$ and the latent variables $U$. In this paper, we propose a data-cloning SMC algorithm as a general purpose optimization routine for MLE of latent variable models, and the algorithm can be treated as a black-box in implementation. Our proposed method shares some important features of the aforementioned two categories of estimation approaches. First, latent variables are marginalized out as in SML. Second, the model parameters are updated via Sequential Monte Carlo (SMC) through a sequence of target distributions generated by data cloning that converge to the sample maximum likelihood estimate. The recent progress in data cloning and SMC is at the core of our proposed method, and we shall review the literature briefly.

---

1Further details regarding SA can be found in Brooks and Morgan (1995).
The idea of data cloning emerges from both biostatistics (Lele, Dennis, and Lutscher (2007); Lele, Nadeem, and Schmuland (2010)) and financial time series (Jacquier, Johannes, and Polson (2007)). Instead of sampling from \( f(\theta, U) \), the above authors propose to sample the joint distribution of the parameter \( \theta \) and \( m \) clones of the data coupled with \( m \) independent sets of latent variables, \( f(\theta, U_1, \ldots, U_m) \).\(^2\) This extended target has a marginal distribution of \( \theta \) that is proportional to the likelihood raised to the power of \( m \).\(^3\) Therefore, when \( m \) is sufficiently large, the target distribution concentrates itself around the sample MLE and “flatten” all local maxima just like SA. Sampling \( f(\theta, U_1, \ldots, U_m) \) is then equivalent to performing optimization.

The key to data-cloning’s success relies on an effective sampler for the extended target \( f(\theta, U_1, \ldots, U_m) \). However, the MCMC-based samplers critically hinge on a good proposal distribution, which requires non-trivial model-specific knowledge in Bayesian statistics. While the SMC algorithm of Johansen, Doucet, and Davy (2008) is more robust than single-chain MCMC methods, its efficiency still critically hinges on using reasonably efficient MCMC kernels over the extended state-space to move particles, which may be hard to come by in general. Furthermore, when dealing with fractional \( m \), even models with a conjugate structure may not have simple proposals. In short, the algorithm of Johansen et al. (2008) still requires non-trivial user input.

To overcome the practical difficulty associated with sampling the joint density of parameters and latent variables, the recent progress on Particle MCMC (Andrieu, Doucet, and Holenstein (2010)), and SMC\(^2\) (Chopin, Jacob, and Papaspiliopoulos (2012), Fulop and Li (2013), Duan and Fulop (2015), among others) suggest that one can build efficient proposals by marginalization. These methods take a two-layer approach, where in the inner layer

\(^2\)The name data-cloning comes from the fact that algorithms require computing the product of \( m \) conditional densities \( \prod_{i=1}^{m} f(Y|U_i) \), where \( Y \) is the observed data. This “cloned” likelihood can be understood as one independently repeats another \( m-1 \) experiments, and by coincidence, all the realizations happen to be \( Y \).

\(^3\)Cloning is a numerical tool to gain computational efficiency, which does not give rise to a legitimate Bayesian interpretation even if a prior distribution is provided. Cloning can be likened to repeatedly updating the prior \( m \) times while the observed data stays fixed. Obviously, only one-time update is legitimate in the Bayesian context.
latent states are marginalized out by importance sampling or particle filtering. Then, the analyst only has to deal with the marginal distribution of the model parameters – a much lower dimensional object. Typically, the paradigm based on marginalization requires much less design efforts compared to traditional approaches targeting the extended state space.

Motivated by these concurrent developments, we devise a data-cloning SMC$^2$ algorithm by incorporating data cloning into the density-tempered SMC$^2$ sampler of Duan and Fulop (2015) with the aim of tackling real-world applications. Specifically, we sequentially sample from a sequence of target marginal distributions that are proportional to the likelihood raised to the power of $m = 1, 2, \ldots$. When $m = 1$ (no cloning), our algorithm boils down to that of Duan and Fulop (2015). To reduce the computational load, we also propose a practical way to adjust the number of latent state particles on the fly. The main advantages of this new algorithm are (1) its relative simplicity for the users and (2) computational efficiency. First, our algorithm can be treated as a black-box that requires little user design effort. For frequentist analysts, the only need is to switch the optimization routine to our proposed method without having to re-code the simulated likelihood already in use. There is no need to carefully choose a proposal distribution either. Second, in contrast to MCMC or SA, our algorithm can be massively parallelized to fully utilize modern parallel hardware like GPUs or clusters.

To begin, we benchmark our method on a linear Gaussian state space model where a closed-form likelihood function is available. As expected, our result shows that as the number of clones increases, the objective value converges quickly and the Monte Carlo variability of the achieved objective value shrinks. Then, we move on to the two important classes of economic/finance problems. The first application is a static mixed-logit model on a fishing mode choice dataset where marginalization of latent variables can be conducted with simulated likelihood. This analysis reveals the power of our algorithm in robustly handling the well-known multi-modality in this class of objective functions that has plagued traditional MLE optimizers. Second, we apply our algorithm to a sample of stock returns on 100 common stocks traded in China where daily price move are subject to a 5 or 10% price limit,
depending on whether a common stock is under special treatment by the stock exchange. The model adopted for these return series is a censored NGARCH process to account for frequent occurrences of hitting daily price limits by Chinese stocks over 2006-2017. Due to price limits, both the underlying stock price and its volatility are generally latent and need a particle filter for the marginalization task. For the Chinese stock data set, we find that ignoring censoring can lead to a substantial underestimation of both the level and variability of the stock volatility.

The remaining sections are organized as follows. Section 2 provides an overview on SMC and data cloning. Section 3 briefly describes the essence and performance of the new algorithm while leaving the technical detail to the appendix. Section 4 presents two applications and their performance on real-world data. Section 5 concludes with additional discussions on simulation techniques with potential to further improve the algorithm’s computational efficiency.

2 An Overview of Data Cloning and SMC

MLE involves maximizing the likelihood function $L(\theta|Y)$, a function of $d$-dimensional parameter vector $\theta$, conditional on the observed data set $Y$. Casting in the Bayesian context, $L(\theta|Y)$ can be interpreted as the posterior distribution corresponding to a constant prior (or improper prior). As a result, one can in principle deploy various simulation techniques to sample $L(\theta|Y)$, and then use the simulated sample mode/mean as the MLE. Among all contenders, we argue that SMC is a preferable method to sample from $L(\theta|Y)$ as it takes advantage of the SMC sample to build a natural proposal to advance the system sequentially in an efficient manner, and the algorithm is also naturally parallelizable. Moreover, the SMC sample constitutes an empirical stationary distribution for the target distribution at every stage, and as opposed to MCMC, the SMC sample updating via the Metropolis-Hastings (MH) move does not hinge on the convergence of the underlying Markov chain.

However, a naive application of SMC in the case of models with latent variables may
not necessarily lead to a good optimization algorithm. First, one needs a good procedure to handle marginalization of latent variables. Second, for the optimization purpose, the likelihood function based on a finite sample may not be the most desirable target, because it could be highly skewed or multi-modal, which makes the simulated sample mode/mean a poor candidate for MLE.

For the first issue, the likelihood function usually has no tractable analytical form and will need to be approximated by \( \hat{L}(\theta|Y, U_{1:p}) \), where \( U_{1:p} \) denotes \( p \) independent sets of random variables used in marginalization. There are basically two types of approximation: (1) importance sampling for static models, and (2) particle filtering for dynamic models, both of which are special cases of SMC. The approach of Duan and Fulop (2015) adopted here takes a two-layer approach. Given a parameter particle \( \theta_i \), the inner loop of the algorithm draws the latent state particles, \( U_{i:p}^{(i)} \), by SMC, independently across \( i = 1, \ldots, n \). The outer loop of the algorithm then updates \( (\theta_i, U_{i:p}^{(i)}) \) towards the posterior marginal \( L(\theta|Y) \) by another SMC. Since two layers of SMC are involved, i.e., one for marginalization and the other for parameter estimation, the algorithm is of SMC\(^2\).

To tackle the second issue, we use data cloning to produce a well-behaved target for optimization. We clone the observed data, \( Y \), \( m \) times, and treat each copy as if it were an independent sample. To obtain the simulated likelihood for each sample (evaluated at a particular \( \theta \)), we further simulate \( m \) independent set of \( p \)-particles \( U_{1:p} \) for marginalization. Formally, we propose to sample from the following extended joint distribution (leaving out a norming constant) conditional on the data:

\[
f^{(m,p)}(\theta, U_{1:mp}|Y) = \prod_{i=1}^{m} \hat{L}(\theta|Y, U_{(i-1)p+1:ip}) \psi(U_{(i-1)p+1:ip}|\theta, Y)
\]

To be specific, if a particle filter is deployed to perform marginalization, \( f^{(m,p)}(\theta, U_{1:mp}|Y) \) is the product of \( m \) independent approximate densities obtained by \( m \) independent runs of particle filtering each with \( p \) particles. Since particle filtering/importance sampling yields
an unbiased estimate of the likelihood, the marginal density for $\theta$ equals

$$
\int_{u_{1:mp}} \left[ f^{(m,p)}(\theta, U_{1:mp} | Y) \right] du_{1:mp} = E_{\psi(\cdot | \theta, Y)} \left[ \prod_{i=1}^{m} \hat{L}(\theta | Y, U_{(i-1)p+1:ip}) \right] = [L(\theta | Y)]^m
$$

This implies that eq (1) targets $[L(\theta | Y)]^m$ with simulation errors in $\hat{L}(\theta | Y, U_{(i-1)p+1:ip})$ for $i = 1, 2, \cdots, m$, and it is an extension of the “exact approximation property” of the pseudo-marginal approach (Andrieu and Roberts (2009)) to the context of data cloning. Note that eq (2) is the same target as in Jacquier et al. (2007) and Johansen et al. (2008). The reason that eq (2) is more preferable than the plain $L(\theta | Y)$ is because when $m$ is large enough, eq (1) according to Lele et al. (2010) eventually behaves like a normal distribution centered at the sample MLE. As a result, the importance of multi-modality, if present in the likelihood function, will gradually fade as $m$ increases, as the density concentrates around the global optimum. This in turn reduces the Monte Carlo variability of $\text{SMC}^2$. In contrast to Jacquier et al. (2007) and Johansen et al. (2008), our method is more efficient and only requires minimal user-input, thanks to the desirable properties of $\text{SMC}^2$ discussed in the introduction.

### 3 Data-Cloning $\text{SMC}^2$ Algorithm

The extended joint distribution in eq (1) induced by data cloning can be sampled by the density-tempered SMC algorithm of Duan and Fulop (2015), because cloning can be treated as needing for density tempering from power of $m$ down as opposed to from the power of 1 down. As already shown in Duan and Fulop (2015), density-tempered SMC is generally more efficient and robust as compared to the expanding-data SMC of Chopin et al. (2012) and Fulop and Li (2013)). Through the exposition below, it will also become clear that due to the unique structure of data cloning, the expanding-data SMC is not really a suitable contender for this purpose.

We develop the algorithm along the line of density-tempered SMC with two important additional adjustments. First, there is no need to initialize the algorithm from the prior
distribution as in Duan and Fulop (2015), because the purpose here is optimization not Bayesian inference, which as discussed in a footnote earlier that Bayesian inference is not valid in the context of data cloning anyway. Second, we propose a procedure to automatically adjust \( p \), recognizing the fact that a large clone number, \( m \), naturally requires a larger number of latent state particles, \( p \), to dampen Monte Carlo errors in the inner-layer SMC.

The technical aspect of our data-cloning SMC\(^2\) algorithm is described in the appendix. The overall structure of the algorithm comprises four parts: (1) initialization sampling, (2) density-tempering sequence and resampling with importance weights, (3) support boosting with the MH move, and (4) automatically adjusting the number of latent state particles.

We now evaluate the performance of our data-cloning SMC\(^2\) algorithm for a simple linear Gaussian state-space model for which exact marginalization can be performed with the Kalman filter. It is therefore possible to turn our algorithm in this case into data-cloning SMC instead of SMC\(^2\). Nevertheless, we forgo the Kalman filter and deploy a particle filter to show the performance of the data-cloning SMC\(^2\) algorithm. This simple test model is

\[
\begin{align*}
    y_t &= \mu + x_t + \sigma_\varepsilon \varepsilon_t \\
    x_t &= \phi x_{t-1} + \sigma_\eta \eta_t \\
    \begin{pmatrix}
        \varepsilon_t \\
        \eta_t
    \end{pmatrix} &\sim N(0, I_{2\times2}) \text{ and } x_0 \sim N\left(0, \frac{\sigma_\eta^2}{1 - \phi^2}\right)
\end{align*}
\]

To ensure positive variances, we parameterize two parameters with a log-transform, i.e., take \( \theta = (\mu, \phi, \ln \sigma_\eta, \ln \sigma_\varepsilon) \). We also restrict \( \phi \in (-1, 1) \) using a truncated sampler for this parameter. For the initialization sampler, we use a Uniform \(( -1, 1) \) for \( \phi \) and a Gaussian for \(( \mu, \ln \sigma_\eta, \ln \sigma_\varepsilon) \) with the mean vector \((0.25, \ln(1.5), \ln(0.475))\) and the covariance matrix \( I_{3\times3} \). We also set \( n = 500 \) parameter particles. The resampling and support-boosting steps are triggered whenever the ESS drops below 50%. After resampling, support boosting will continue until the average cumulative acceptance rate achieves 100%. We use independent multivariate normal moves with its parameters being estimated using the SMC sample at that stage. For the linear Gaussian state-space model, the locally optimal proposal \( f(x_{t+1} | \)
$(x_t, y_{t+1})$ is available analytically, and it is what we use to sample in the adaptive particle filter. In what follows, we deploy stratified resampling within the particle filter.

We generate one data set of size $T = 500$ observations using parameter values $\theta^* = (0.5, 0.825, \ln(0.75), \ln(1))$. We then implement 50 independent runs of the data-cloning SMC$^2$ algorithm, beginning with $p = 50$ latent state particles, automatically adjusting $p$ as described in Section A.4 and running up to $m = 20$ clones. Figure 1 displays the results of these 50 runs. The upper panel shows the log-likelihood values at the parameter estimates obtained by the algorithm from the 50 runs where the exact log-likelihood function is plotted (using the Kalman filter to evaluate at the parameter value). While there exists some variability for the un-cloned objective function value (i.e., $m = 1$), all runs quickly converge to the same functional value as $m$ increases. The middle panel shows the average number of latent state particles that the algorithm automatically deploys to ensure the acceptance rate above a pre-specified level of 20%. As expected, the number of latent state particles deployed, $p$, is roughly linear in $m$. Lastly, the lower panel shows the mean runtime of the algorithm. Table 1 further reports the mean and standard deviation of the log-likelihood values and the average runtime for this model.\(^4\) Evidently, cloning ensures parameter estimates little affected by different initialization distributions.

\section{Two Applications}

We consider (1) a mixed-logit specification for modeling decision making and (2) a censored GARCH model to describe returns on common stocks traded in China where prices are subject to daily either 10 or 5% up/down price limits, depending on whether a common stock is under special treatment. Both models involve latent variables and have wide applicability in practice. For mixed-logit models, it is relatively straightforward to perform marginalization, whereas the censored GARCH model is more complex with a non-linear and non-Gaussian dynamic structure that requires a particle filter to estimate its log-likelihood.

\(^4\)We use Matlab GPU-enabled functions running on a Nvidia P100 GPU for the likelihood calculation for all the models considered in this paper. Porting the code to CUDA would yield substantial speedup.
A good MLE algorithm should have at least two properties: robustness and generality. By robustness, we mean that the algorithm can produce reliable results with little changes across different starting points and tuning parameters. By generality, we mean that the algorithm is applicable to a wide range of models. We will demonstrate robustness of the data-cloning SMC\(^2\) algorithm through a mixed-logit model for decision making, and generality through estimating a censored GARCH model to factor in the effect of price limits on common stocks traded in China.

### 4.1 Discrete choice – a mixed-logit model

It has been well documented that the mixed-logit likelihood, in particular with non-normal mixing distribution, is difficult to obtain the MLE, and hence the choice of the starting value becomes crucial. Here, we compare the performance of our data-cloning algorithm with a standard gradient-based optimization routine using a real decision making dataset.

Suppose there are \(N\) individuals making discrete choices over \(J\) alternatives (assuming no outside options). The individual \(i\)'s conditional choice probability for alternative \(j\) is given by

\[
L_{ij}(\beta) = \frac{\exp(\beta' x_{ij})}{\sum_{k=1}^{J} \exp(\beta' x_{ik})},
\]

where \(x_{ij}\) is a set of variables containing individual-specific characteristics (for example, age or income) as well as the alternative-specific attributes (for example, the cost of taking train vs. plane), and \(\beta\) is unknown and distributed according to \(\phi(\beta|\theta)\) where \(\theta\) is the hyperparameter of interest.

Since likelihood functions with multiple modes are often associated with small samples, we take the first 100 observations (i.e., \(N = 100\)) from the fishing mode choice dataset illustrated in Cameron and Trivedi (2005). There are four fishing modes (i.e., \(J\) has four elements):

\(^5\)For state-space models in general, users only need a reasonably efficient particle filter to apply our method.
beach, pier, private boat, and charter boat. Each choice mode is characterized by two alternative-specific regressors: the fishing cost and catch rate. In practice, researchers often assume the mixing distribution $\phi(\beta|\theta)$ to be normal. However, the normality assumption will imply that for some individuals, the response coefficient to price is positive, which contradicts the classical demand theory and thus may lead to a wrong conclusion. To address this issue, one can impose a sign restriction by assuming, for example, that the price response coefficient follows a log-normal distribution. However, it has been documented that mixed-logit models with log-normal components tend to experience numerical problems in estimation. For example, Hensher and Greene (2003) stated that “experience suggests that they iterate many times looking for the maximum, and often get stuck along the way”, and Train (2009) reached a similar conclusion. Here we assume that the coefficient of fishing cost follows a log-normal distribution to respect the classical price theory. For the coefficient of catch rate, we let it follow a normal distribution. For simplicity, these two distributions are further assumed to be independent.

Since the analyst does not observe $\beta$, it has to been marginalized out in order to obtain the unconditional choice probability $P_{ij} = \int L_{ij}\phi(\beta|\theta)d\beta$, which can be approximated by an unbiased estimator:

$$\hat{P}_{ij} = \frac{1}{p} \sum_{r=1}^{p} L_{ij}(\beta^{(r)})$$

where $p$ is the number of particles used in marginalization. The simulated log-likelihood is then given by

$$\sum_{i=1}^{N} \sum_{j=1}^{J} d_{ij} \log \hat{P}_{ij}$$

with $d_{ij} = 1$ if individual $i$ chooses alternative $j$ and 0 otherwise.

For the classical simulated MLE, we use 10,000 Halton sequence for each individual.\(^6\) It is worth noting that the objective function obtained by applying a Halton sequence can be understood as a numerical integration with 10,000 grid points spaced out with a high

\(^6\)The Halton sequence is generated from Matlab by skipping the first 1,000 values, and then retain every 101st point.
degree of uniformity. Since it is an approximate objective function, the maximum of the approximate objective function is naturally not the true MLE of the sample. Nevertheless, it is a differentiable function of the model parameters, which in turn permits gradient-based optimization. We perform MLE for each of 50 randomly generated starting values, which are generated from a multivariate normal distribution.

In the case of the data-cloning SMC\textsuperscript{2} algorithm, we fix the number of parameter particles at $n = 500$ and update the tempering parameter $\delta$ such that the resulting ESS approximately equals 50\%, which is then followed by resampling and support boosting. We set the cumulative acceptance rate in the MH move to 100\% and to move the particles we use a normal mixture proposal with four components fitted on the existing particle set. We generate 50 sets of initialization samples with their means set equal to the 50 starting values used in the classical MLE and the covariance matrix equal to $3 \times I_{4 \times 4}$. The scale parameter for each of the two mixing distributions is parameterized by a log-transformation to ensure positiveness.

We execute the data-cloning SMC\textsuperscript{2} algorithm under different combinations of latent state particles, $p$, and cloning copies, $m$. Following Johansen et al. (2008), we report in Table 2 the summary statistics of the log-likelihood (objective function) at the optimal solution\textsuperscript{7} and the average computing time. The wide range of solutions for 50 random starting values suggests that the classical MLE method is very sensitive to the starting value, which is consistent with the existing literature cited earlier. In contrast, even the SMC\textsuperscript{2} without data cloning already outperforms the classical MLE method on average in terms of accuracy. Further, the best solution without data cloning over the 50 initialization samples under $p = 200$ equals -120.38, which is only slightly worse than the best solution found by the classical MLE method, i.e., -120.11.\textsuperscript{8} When the number of clones, $m$, increases, accuracy also improves (in terms

\textsuperscript{7}Unlike the typical MLE method that maximizes a non-random objective function, SMC\textsuperscript{2} can handle a stochastic objective function arising from sampling the latent state particles in marginalization. The objective function in the mixed-logit model is quasi-random due to the use of the Halton sequence approximation. For a more meaningful comparison, we thus plug the SMC\textsuperscript{2} estimate into the MLE objective function that uses the same Halton sequence.

\textsuperscript{8}As described earlier, the classical MLE obtained for the approximate log-likelihood is not the true MLE of the data sample. Hence, it is conceivable that the true MLE based on the true objective function may yield a value lower than -120.11.
of either mean or maximum). But the main improvement delivered by the data-cloning SMC\(^2\) algorithm appears to be its robustness: for example, when \(m = 50\) and \(p = 200\), the standard deviation of the estimate generated by the data-cloning SMC\(^2\) algorithm is closed to zero across different initialization distributions. Table 2 highlights that neither the initialization distribution nor the tuning parameter seems to exert material impact on the data-cloning SMC\(^2\) algorithm when a reasonable number of clones are employed. We contend that robustness is of the first-order importance to empirical analysts, and the data-cloning SMC\(^2\) algorithm appears to have met this requirement.

Regarding the computing time, the classical MLE method might appear to be more computationally efficient if just being run once, judging from the average computing time reported in Table 2. However, the total amount of time spent on experimenting with different starting values in addressing multi-modality could still be substantial, which in this case equals \(50 \times 37 = 1850\) seconds. By contrast, the data-cloning SMC\(^2\) algorithm does not require an extensive search for initial values, and hence is computationally attractive. In the most computationally demanding case reported in Table 2, i.e., \((m = 100, p = 200)\), it only takes about 674 seconds to complete one run, and the algorithm only needs one run.

Table 2 also reveals an interesting and practical issue in smartly choosing \(p\). When \(m = 100\), small number of latent state particles \(p = 200\) actually requires more support-boosting steps, and hence is less efficient than \(p = 400\). Since it is difficult to gauge an optimal \(p\) ex ante, we have suggested in Section A.4 a procedure to automatically adjust \(p\) on the fly. We show how it performs through the following experiment: set the initial number of the latent state particles \(p\) to 50, double \(p\) whenever the MH acceptance rate drops below 20\%, and terminate the algorithm at \(m = 100\). The upper panel of Figure 2 reports the log-likelihood as a function of \(m\) for the 50 runs using the same 50 random starting initialization distributions described earlier. Some variations are visible at \(m = 1\), but differences across 50 runs disappear very quickly after reaching a moderate level of cloning at \(m = 10\). The middle panel plots the average number of latent state particles chosen by the algorithm at different \(m\), and the lower panel plots the average run time. For example, the algorithm
determines that $p = 400$ is sufficient when $m$ is greater than 80.

### 4.2 Stocks subject to price limits – a censored GARCH model

Censoring due to daily price limits is a critical feature that should be factored in when one models returns on stocks traded in the Chinese exchanges (Shanghai and Shenzhen stock exchanges) and some other jurisdictions. Here we focus on the so-called A shares traded in Chinese exchanges denominated in RMB. Price limits imposed by the regulators in China are 5 or 10% per day, depending on whether a stock on a particular trading date is under special treatment (ST).\(^9\) Table 3 report the number of limit hits in the period of 2006-2017. One can see that the phenomenon is widespread with a large number of stocks experiencing some limit violations in every year. Furthermore, hitting price limits is, generally speaking, a cyclical phenomenon, with many more occurrences in more volatile years such as 2015 or 2016.

As a concrete example, Guanghui Logistics Co. frequently hit price limits over the time period of 2006-2008 as evidenced in the top panel of Figure 3. One obvious implication of ignoring the fact that reported prices are sometimes censored is the downward distortion in estimated stock volatility regardless of the model (dynamic or static). This simply reflects that the reported prices are the intrinsic prices being censored on either side if they exceed the price limit and thus exhibit less volatility. Naturally, more frequently censored stocks will experience more severe distortions, and more volatile stocks will also face more significant downward biases in estimation.

Two factors are critical to modeling stock returns subject to price limits – accurately capture the censoring mechanism and adequately reflect the underlying price dynamic. Our adopted model is a Tobit-style specification where the reported censored price, $P_t$, is linked

\(^9\)ST is the designation assigned by the exchanges to those listed stocks that in the view of the exchanges are experiencing abnormal conditions and hard for investors to assess the companies’ future.
to a partially observed intrinsic price, $P_t^*$, through the following relationship:

$$
P_t = \begin{cases} 
P_t^* & \text{if } d_t < \frac{P_t^*}{P_{t-1}} - 1 < u_t \\
P_{t-1}(1 + d_t) & \text{if } \frac{P_t^*}{P_{t-1}} - 1 \leq d_t \\
P_{t-1}(1 + u_t) & \text{if } \frac{P_t^*}{P_{t-1}} - 1 \geq u_t 
\end{cases}$$

where $u_t = 10\%$ and $d_t = -10\%$ are the upper and lower censoring points defined on daily returns for all common stocks except when a stock is classified as ST. For the ST stocks, $u_t = 5\%$ and $d_t = -5\%$. A common stock may go in and out of ST, and therefore we use subscript $t$ to reflect changes to the status over time.

Given the success of the GARCH model in modeling stock returns widely reported in the literature, we assume that log-returns defined on the intrinsic stock prices follow the non-linear GARCH(1,1) (NGARCH) model of Engle and Ng (1993). The NGARCH model permits asymmetric volatility responses to return shocks and is, for example, able to parsimoniously capture the well-known leverage effect. The NGARCH model can be stated as:

$$
\ln \frac{P_t^*}{P_{t-1}} = \mu + \sqrt{h_t} \varepsilon_t 
$$

$$
h_t = \alpha_0 + \alpha_1 h_{t-1} + \alpha_2 h_{t-1}(\varepsilon_{t-1} - \theta)^2 
$$

$$
\varepsilon_t \sim GED(\beta) \quad \text{i.i.d. over } t
$$

Using the generalized error distribution is to allow for conditional fat-tails, and $\beta$ is its parameter governing tail fatness. In addition, we impose the usual sufficient conditions for positive variances and variance stationarity: $\alpha_0 > 0$, $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, and $\alpha_1 + \alpha_2(1+\theta^2) < 1$.

Prior to the first occurrence of censoring, $P_t^* = P_t$ and the stochastic variance, $h_t$, is observable by deducing from eq (10). Once the price limit is hit, the stochastic variance becomes forever latent even if subsequent stock prices no longer hit the price limit. Consequently, the likelihood function is not available in a workable closed form as long as the price limit has been hit once. However, it is reasonably straightforward to develop an adaptive particle filter for this model because the one-period predictive distribution $f(P_t \mid h_{t-1}, P_t^*, P_{t-1})$ is available in closed form and sampling from the conditional distribution $f(\varepsilon_t \mid h_{t-1}, P_t^*, P_{t-1})$.
can be easily conducted. In short, we are able to generate efficient and unbiased estimates of the likelihood by a particle filter and implement our marginalized data-cloning method to obtain the maximum likelihood estimate.

A further technical complication arises from the fact that trading may be temporarily suspended for various reasons (for example, a pending merger/acquisition) and stock prices may be missing for some dates. We address this by assuming that as long as the gap stays under some preset number of trading days, $N_{\text{MaxGap}}$, the system with missing data continues to follow the same underlying dynamics over the gap. Otherwise, we allow for a structural break and treat the data after the gap as from a new common stock. In our implementation, $N_{\text{MaxGap}}$ is set to 5 trading days, i.e., one calendar week. Let us now briefly describe how the former case is handled in particle filtering. Denote the last time point before the gap by $t_i$ and the subsequent resuming time point by $t_i+1$ where the gap is no larger than 5 trading days. We simply simulate the system forward to $t_i+1$ using the transition density starting from the filtering distribution at $t_i$. To control the Monte Carlo error, we temporarily increase the number of particles $M_{\text{rep}}$ times.\footnote{Our experience suggests that this simple sampling scheme works well enough for our purpose. With a larger $N_{\text{MaxGap}}$, one can expect the simple sampling method of blindly going forward would perform poorly, because it is not guided by the resurfaced stock price at $t_i+1$. In such cases, one can resort to the non-Gaussian bridge sampler of Duan and Zhang (2016) to increase sampling efficiency.}

We run our SMC routine over a transformed parameter space to ensure that the parameter constraints are automatically satisfied: $\mu = \mu$, $\kappa_0 = \ln(\alpha_0)$, $\kappa_1 = \text{Logit}^{-1} (\alpha_1 + \alpha_2 (1 + \theta^2))$, $\kappa_2 = \text{Logit}^{-1} \left( \frac{\alpha_1}{\alpha_1 + \alpha_2 (1 + \theta^2)} \right)$, $\theta = \theta$, and $\tilde{\beta} = \text{Logit}^{-1} (\beta/2)$. The proposal sampler in the support-boosting step is the result of fitting a four-component mixture normal model to the SMC sample of parameter particles at that stage. We run our algorithm up to $m = 10$ clones with $n = 1,000$ parameter particles, and initialize the algorithm with $p = 100$ state particles.

We take 100 Chinese firms experiencing most price limit violations. For each of these 100 firms, we keep the longest contiguous data sample, allowing for a maximum gap of 5 trading
days, i.e., $N_{\text{MaxGap}} = 5$. Over the gap, we quintuple the number of latent state particles, i.e., $M_{\text{rep}} = 5$. The resulting data series for these 100 Chinese stocks have an average length of 1,143 days per stock, and on average 9.7% of the daily returns hit the price limits. The average runtime of the data-cloning SMC$^2$ algorithm is 261 minutes per firm. To fit the GPU memory, we have bounded the number of latent state particles at $p = 6,400$. The average number of latent state particles across the 100 firms at the completion of the algorithm is $p = 561$.

We also estimate the uncensored NGARCH(1,1) model on these 100 stock return series to better appreciate the effect of censoring on the variance estimates. For the uncensored model, it is a data-cloning SMC algorithm instead of SMC$^2$, because in the absence of latent state the likelihood value can be evaluated with a closed-form formula and particle filter is no longer needed.

Figure 3 reports the variance estimates for Guanghui Logistics Co. from both the censored and uncensored NGARCH(1,1) models. The upper panel displays its observed returns over the period from 2006 to 2008 with the red dots pointing out the times when the 5% limit was hit. The lower panel shows the two variance paths corresponding to censored and uncensored NGARCH models. Initially, the returns stay within the price limits, and the two models provide very close variance estimates. At around the third quarter of 2006 volatility began to increase and the price limit started to bind. From that point onwards, the variance estimates from the two models diverge sharply with the variance path under the censored NGARCH(1,1) model being substantially higher and much more variable. Intuitively, when the price limit is hit, the variance innovation should have been driven by a larger shock to the unobserved intrinsic price but instead by a smaller and wrongly inferred shock when the uncensored model is used. The distortion will be particularly severe when censoring is applied to a shock drawn from the deep tails. Once a distortion to variance occurs, it will affect the next return shock leading to an avalanche-type divergence between the two variance paths over 2007 as illustrated in Figure 3. Ignoring censoring obviously causes a substantial underestimation of both the average level and the variability of the stochastic
To ascertain that such distortions are systematic and quantitatively significant, we present in Table 4 some statistics obtained for our sample of 100 firms. We group these 100 firms into three terciles by the frequency of hitting price limits and report the quantities of interest for each tercile in separate rows. The first column is the average market capitalization in billions RMB for each group. The second column shows the fraction of returns that hit the price limit. The third column provides the category average of the ratio of the time series average of the stochastic variance for each firm under one model over that of another model (censored vs uncensored). Finally, the fourth column presents the category average of the time series variance of the stochastic variance for each firm under one model over that of the another model (censored vs uncensored). The ratio on the average variances implied by the two models clearly reveals a serious downward bias resulting from the failure to take care of censoring, and the effect ranges from 90% for the top tercile to 20% for the bottom tercile. Furthermore, the results in column four suggest that the bias on the variance of stochastic variance is even more substantial, with the censored model yielding many times higher variance of variance even for the firms in the bottom tercile. Overall, these results suggest that whenever price limits are recurrent, their effect can be substantial on the estimation of variance.

Finally, Table 5 reports the average parameter estimates with and without censoring in the first two columns. The most striking result is that ignoring censoring leads to substantially higher estimates for the GED shape parameter, i.e., $\beta$, leading to a conclusion of thinner tails. Columns 3, 4 and 5 provide standard deviation, maximum and minimum differences in the parameter estimates with and without censoring and these statistics are computed with our sample of 100 Chinese firms. Evidently, we can observe quite a bit of divergence of the parameter estimates between the two models. The persistence parameter $\alpha_1$ is in particular significantly larger for some firms when censoring is accounted for.
5 Discussion and Conclusion

We propose a generic data-cloning SMC\textsuperscript{2} algorithm to compute the MLE for models with latent variables. Our optimization method is fairly easy to implement for a wide range of applications where importance sampling or particle filtering can be used to marginalize out latent variables. Our data-cloning SMC\textsuperscript{2} algorithm is shown to work very well in two applications in economics/finance. We now conclude by discussing a few strategies for further improvements in the aspect of computational efficiency.

One challenge associated with data cloning is its inevitable need for increasing the number of latent state particles, $p$. Otherwise, the MH move’s acceptance rate in the support-boosting step will significantly drop once going beyond some number of clones. We have proposed an automated procedure for choosing $p$. Apart from this simple and intuitive strategy, two important progresses in the literature may also offer a practical solution. First, one can deploy randomized low discrepancy sequences for marginalization. Due to their higher uniformity, it will generally require a lower $p$. This approach has been widely adopted for cross-sectional models such as mixed-logit/multivariate probit, and in fact we have already implemented in the first example involving a mixed-logit model. Naturally, it can also be applied on dynamic models, thanks to the pioneering work of Gerber and Chopin (2015) on sequential quasi-Monte Carlo. Another source for low acceptance rates comes from the non-smoothness of the pseudo-marginal approach of Andrieu and Roberts (2009) and Andrieu et al. (2010). The acceptance probability is mainly determined by the likelihood ratio, $f_{\delta,m}(\theta^*, U_{1:mp}^*|Y)/f_{\delta,m}(\theta, U_{1:mp}|Y)$. Since $U^*$ and $U$ are independent, the discrepancy between these two likelihoods may be large even when $\theta^*$ and $\theta$ are close to each other. Recently, Deligiannidis, Doucet, and Pitt (2016) proposed a “correlated pseudo-marginal” method to mitigate this problem. Their suggestion is to couple $U^*$ and $U$ using a simple AR(1) process: $U^* = \rho U + \sqrt{1-\rho^2}\epsilon$, where $0 < \rho < 1$ and $\epsilon$ is normally distributed with zero mean and identity covariance matrix. This approach enjoys some level of smoothness similar to deploying common random numbers while maintaining unbiasedness of the pseudo-marginal approach.
Lastly, our data-cloning SMC$^2$ algorithm is naturally suited for massive parallelization. With $m$ clones, $p$ latent state particles and $n$ parameter particles, one can run $n$ independent importance sampler/particle filter (each with $m$ clones and $p$ latent state particles) in parallel. As our examples have illustrated, modern inexpensive parallel hardware (e.g., GPU) can be easily deployed to speed up the algorithm.

A Technical description of the data-cloning SMC$^2$ algorithm

A.1 Initialization sampling

When applying the SMC-type samplers in Bayesian inference, such as Del Moral, Doucet, and Jasra (2006), Chopin et al. (2012), Fulop and Li (2013) and Duan and Fulop (2015), one naturally starts with an initial set of parameter values by sampling from the prior distribution. However, initialization sampling need not be tied to the prior belief, especially when the primary goal is optimization. We adopt the idea of Duan and Wang (2017) to remove the impact of the initialization sampler from the system so that the initialization distribution does not inadvertently become a prior belief. Let $I(\theta)$ be an arbitrary density that generates $\theta$. It should be noted that the only technical requirement is that $I(\theta)$ must have a support no smaller than the target distribution. Different initialization samplers only change the efficiency of the SMC algorithm. We obtain an initial sample of size $n$ from $I(\theta)$, i.e., $\{\theta_i; i = 1, 2, \cdots, n\}$.

A.2 Density-Tempering Sequence and Resampling with Importance Weights

With the initial sample, one can attach importance weights to different sample points to obtain an empirical distribution for $f^{(m,p)}(\theta, U_{1:mp}|Y)$. However, the importance weights are likely to be quite uneven, making the empirical distribution of $\theta$ a poor proxy to $f^{(m,p)}(\theta, U_{1:mp}|Y)$. The uneven importance weights, however, can be tempered through a sequence of steps so that the final empirical distribution becomes a representation of the
desired target density.

The tempering sequence of intermediate target density functions is structured in the following way:

\[
f_{\delta,m}(\theta, U_{1:mp}|Y) \propto \begin{cases} 
\left( \frac{\mathcal{L}(\theta|Y, U_{1:p})}{I(\theta)} \right)^{\delta} I(\theta)\psi(U_{1:p}|\theta, Y) & \text{if } m = 1 \\
\frac{f_{1,m-1}(\theta, U_{1:(m-1)p}|Y) \left( \hat{\mathcal{L}}(\theta|Y, U_{(m-1)p+1:mp}) \right)^{\delta} \psi(U_{(m-1)p+1:mp}|\theta, Y)}{f_{\delta,m}(\theta, U_{1:mp}|Y)} & \text{if } m > 1
\end{cases}
\]

For a given value of \( m \), \( f_{1,m}(\theta, U_{1:mp}|Y) \) equals \( f^{m:p}(\theta, U_{1:mp}|Y) \). Through a sequence of intermediate targets, \( 0 < \delta_1 < \delta_2 < ... < 1 \), one can arrive at the final target density under each \( m \) while controlling the importance weights along the way. By starting from \( m = 1 \), one can also advance the target density to reach any desired value of \( m \). Evidently from the above equations, \( I(\theta) \) does not affect the outcome when \( \delta = 1 \), an insight of Duan and Wang (2017), but it can alter the process of getting to the final target. In fact, a good initialization sampler will serve to temper the importance weight and thus reduce the number of tempering steps needed. Unlike changing the prior distribution which changes the target distribution as in Duan and Fulop (2015) and others, \( I(\theta) \) does not alter the target distribution at all, and knowledge from any previous run can be incorporated to improve the performance of the algorithm. This approach naturally works in the Bayesian context by simply targeting the posterior distribution instead of the likelihood function.

The design of this intermediate target in eq (11) is motivated by Duan and Fulop (2015) so that \( \prod_{i=1}^{m} \psi(U_{(i-1)p+1:ip}|\theta, Y) \), a term which is difficult to evaluate, can be completely avoided. Moving from \( \delta_{i-1} \) to \( \delta_i \) can be accomplished by re-weighting the parameter by the following weight:

\[
W(\delta_{i-1}, \delta_i, \theta, U_{1:mp}|Y) = \frac{f_{\delta_i,m}(\theta, U_{1:mp}|Y)}{f_{\delta_{i-1},m}(\theta, U_{1:mp}|Y)} \cdot \begin{cases} 
\left( \frac{\mathcal{L}(\theta|Y, U_{1:p})}{I(\theta)} \right)^{\delta_i-\delta_{i-1}} & \text{if } m = 1 \\
\left( \frac{\hat{\mathcal{L}}(\theta|Y, U_{(m-1)p+1:mp})}{\delta_i-\delta_{i-1}} \right)^{\delta_i-\delta_{i-1}} & \text{if } m > 1
\end{cases}
\]
Following Del Moral et al. (2006) and Duan and Fulop (2015), we choose the next point in the tempering sequence, i.e., \( \{\delta_i\} \), such that the effective sample size (ESS) equals a preset threshold level, say, 50% of the sample size \( n \). As 50% of the sample for \( \theta_i \) depleted through importance weighting, resampling and support boosting\(^{11}\) are required to maintain sample quality as it is well known in the SMC literature. Resampling is straightforward and interested readers are referred to Duan and Fulop (2015) for details. Below we focus on the support-boosting step.

A.3 Support boosting

After resampling, the SMC sample becomes equally-weighted and correctly represents the intermediate target density. However, the empirical support has shrunk, i.e., fewer distinct parameters in the sample of size \( n \). Boosting the empirical support becomes necessary, and can be accomplished by running MH moves. These MH moves are meant to increase the ESS as opposed to getting convergence in the typical MCMC run; see Chopin (2002).

Let \( P_{\delta_i,m}(\theta) \) be the independent proposal density, which we use to generate new parameter value \( \theta^* \), at the stage of \((\delta_i, m)\). A good proposal can be easily constructed by first fitting a normal mixture model to the re-sampled parameter particles, and then use the fitted density as the proposal. The overall proposal density in the extended space is \( P_{\delta_i,m}(\theta) \prod_{i=1}^m \psi(U_{(i-1)p+1:p} | \theta, Y) \). The MH acceptance probability in the extended space for \((\theta^*, U^*_{1:mp})\) at this stage of \((\delta_i, m)\) is:

\(^{11}\)This is also known as “resample-move” in Gilks and Berzuini (2001).
\[ a_{\delta, m} \left\{ (\theta, U_{1:mp}) \rightarrow (\theta^*, U_{1:mp}^*) \right\} = \min \left( 1, \frac{f_{\delta, m}(\theta^*, U_{1:mp}^*)}{f_{\delta, m}(\theta, U_{1:mp}|Y)} P_{\delta, m}(\theta^*) \prod_{i=1}^{m} \psi(U_{(i-1)p+1:ip}|\theta^*, Y) \right) \]

\[
= \begin{cases} 
\min \left( 1, \frac{\delta_i(I(\theta^*))}{I(\theta)} P_{\delta, 1}(\theta) \right) & \text{if } m = 1 \\
\min \left( 1, \frac{\delta_i(I(\theta^*))}{I(\theta)} P_{\delta, 1}(\theta^*) \right) & \text{if } m > 1
\end{cases}
\]

From the standard result, \( f_{\delta, m}(\theta, U_{1:mp}|Y) \) is known to be the stationary density of the Markov kernel defined by the above acceptance probability. Since the sample of \( (\theta, U_{1:mp}) \) already represents \( f_{\delta, m}(\theta, U_{1:mp}|Y) \), the MH move produces a sample that also represents the same distribution on the extended space. It should be emphasized that the purpose of the MH move is only to boost the empirical support of the target distribution and to decrease the Monte Carlo error. More MH moves cannot do damage, but require more computing time. We thus use a self-adaptive criterion to decide on how many MH moves to take. Each MH move on a sample yields an acceptance rate, and we take the MH move until the cumulative acceptance rate exceeds, say 100%, i.e., each parameter has on average been replaced once. After support boosting, one returns to the density-tempering step to advance \( \delta \) to the next suitable value, and then perform resampling and support boosting again. The iterative process continues until \( \delta = 1 \) is reached for every \( m \). When the algorithm terminates, report the sample mean \( \bar{\theta} \) as the MLE.

### A.4 Automatically adjusting the number of latent state particles

While the data-cloning SMC\(^2\) algorithm described above targets \([L(\theta|Y)]^m\), irrespective of the simulation noise in the likelihood estimate, it is now well known in the literature that efficiency of the pseudo-marginal approach ultimately depends on the Monte Carlo noise (see for example Doucet, Pitt, Deligiannidis, and Kohn (2014)). In practice, if the Monte Carlo noise becomes too large, the acceptance probabilities in the MH move will drop towards zero.
Doucet et al. (2014) showed that for particle MCMC the asymptotically optimal number of latent state particles, \( p \), should be chosen such that the standard deviation of the noise of the target density is around 1. In the case of data cloning, it seems very hard to derive a parallel result for a fixed optimal \( p \), because as \( m \) gets larger, the SMC routine moves from less noisy densities (low \( m \)) to gradually more noisy densities. Therefore, we suggest adjusting \( p \) on the fly as the algorithm progresses. In particular, whenever the average acceptance rate drops below a predefined number \( C \), we re-initialize the SMC algorithm by simply doubling the number of latent state particles, i.e., \( p \). This can be implemented by fitting a new initialization distribution \( \tilde{I}(\theta) \) to the existing set of parameter particles. Then, we re-initialize the SMC routine to again target \( f_{\delta,m}(\theta, U_{1:mp}|Y) \) in eq (11) but with \( \tilde{I}(\theta) \) in place of \( I(\theta) \). When \( \delta \) reaches 1, the target density under the new and larger \( p \) is obtained.

References


24


Figure 1: The effect of data cloning for the linear Gaussian state-space model using the data-cloning SMC\(^2\) algorithm based on 50 random initialization distributions

Table 1: Summary statistics for the linear Gaussian state-space model using the data-cloning SMC\(^2\) algorithm based on 50 random initialization distributions

<table>
<thead>
<tr>
<th># of Clones</th>
<th>Log-likelihood Mean</th>
<th>Log-likelihood Std Dev</th>
<th>Runtime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-867.0754</td>
<td>0.042511</td>
<td>57.68</td>
</tr>
<tr>
<td>5</td>
<td>-866.9513</td>
<td>0.003163</td>
<td>525.23</td>
</tr>
<tr>
<td>10</td>
<td>-866.9481</td>
<td>0.000918</td>
<td>1204.95</td>
</tr>
<tr>
<td>20</td>
<td>-866.9472</td>
<td>0.000478</td>
<td>2973.60</td>
</tr>
</tbody>
</table>
Table 2: Summary statistics of the log-likelihood values for the mixed-logit model evaluated at the obtained MLE based on 50 random starting values or initialization distributions

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Runtime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical MLE</td>
<td>-122.89</td>
<td>8.41</td>
<td>-148.68</td>
<td>-120.11</td>
<td>36.91</td>
</tr>
<tr>
<td>SMC²</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of Clones</td>
<td># of Particles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>p=200</td>
<td>-121.62</td>
<td>0.89</td>
<td>-123.88</td>
<td>-120.38</td>
</tr>
<tr>
<td>10</td>
<td>p=200</td>
<td>-120.19</td>
<td>0.04</td>
<td>-120.44</td>
<td>-120.15</td>
</tr>
</tbody>
</table>
| 50            | p=200  | -120.13 | 0.01    | -120.20 | -120.12           | 123.74| 400.15
| 100           | p=200  | -120.12 | 0.00    | -120.13 | -120.12           | 674.06| 900.23
| 1             | p=400  | -121.70 | 1.09    | -125.31 | -120.37           | 6.16  |
| 10            | p=400  | -120.21 | 0.12    | -120.84 | -120.15           | 23.25 |
| 50            | p=400  | -120.14 | 0.06    | -120.50 | -120.12           | 131.23| 400.15
| 100           | p=400  | -120.13 | 0.05    | -120.44 | -120.12           | 436.42| 900.23

Figure 2: The effect of data cloning for the mixed-logit model using the data-cloning SMC² algorithm based on 50 random initialization distributions
Table 3: Summary statistics on price limit violations for all Chinese stocks (A shares)

<table>
<thead>
<tr>
<th>Year</th>
<th># of A share firms</th>
<th># of firms hitting limit</th>
<th># limit hits/# firms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>1367</td>
<td>955</td>
<td>2.67</td>
</tr>
<tr>
<td>2007</td>
<td>1481</td>
<td>1376</td>
<td>7.63</td>
</tr>
<tr>
<td>2008</td>
<td>1545</td>
<td>1239</td>
<td>5.65</td>
</tr>
<tr>
<td>2009</td>
<td>1649</td>
<td>1295</td>
<td>4.82</td>
</tr>
<tr>
<td>2010</td>
<td>1994</td>
<td>1524</td>
<td>3.58</td>
</tr>
<tr>
<td>2011</td>
<td>2278</td>
<td>1443</td>
<td>2.65</td>
</tr>
<tr>
<td>2012</td>
<td>2435</td>
<td>1721</td>
<td>3.19</td>
</tr>
<tr>
<td>2013</td>
<td>2452</td>
<td>1935</td>
<td>3.38</td>
</tr>
<tr>
<td>2014</td>
<td>2577</td>
<td>2120</td>
<td>3.66</td>
</tr>
<tr>
<td>2015</td>
<td>2799</td>
<td>2796</td>
<td>23.48</td>
</tr>
<tr>
<td>2016</td>
<td>3027</td>
<td>2798</td>
<td>6.95</td>
</tr>
<tr>
<td>2017</td>
<td>3460</td>
<td>2399</td>
<td>3.83</td>
</tr>
</tbody>
</table>

Figure 3: The effect of censoring on variance estimates: censored vs uncensored NGARCH(1,1) model for Guanghui Logistics Co. daily over 2006-2008
Table 4: The effect of censoring on variance estimates: censored vs uncensored NGARCH(1,1) model for the 100 Chinese stocks (A shares) that most frequently hit price limits over 2006-2017. Sorting is according to the frequency of hitting price limits.

<table>
<thead>
<tr>
<th></th>
<th>Avg. Market Cap (Billions RMB)</th>
<th>Proportion Censoring</th>
<th>Avg. Ratio Avg. Ratio of ( \text{Var of } h_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top tercile</td>
<td>2.54</td>
<td>15%</td>
<td>1.68</td>
</tr>
<tr>
<td>Mid tercile</td>
<td>4.04</td>
<td>8%</td>
<td>1.39</td>
</tr>
<tr>
<td>Bottom tercile</td>
<td>4.75</td>
<td>5%</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 5: The effect of censoring on parameter estimates: censored vs uncensored NGARCH(1,1) model. Average parameter values and other statistics on differences (between censored and uncensored models) are based on the 100 Chinese stocks (A shares) that most frequently hit price limits over 2006-2017.

<table>
<thead>
<tr>
<th></th>
<th>Censored NGARCH(1,1)</th>
<th>NGARCH(1,1)</th>
<th>Std(diff)</th>
<th>Min(diff)</th>
<th>Max(diff)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_0 \times 10^{-5} )</td>
<td>4.11</td>
<td>4.63</td>
<td>3.04</td>
<td>-20.58</td>
<td>4.29</td>
</tr>
<tr>
<td>( \alpha_1 )</td>
<td>0.88</td>
<td>0.85</td>
<td>0.06</td>
<td>-0.06</td>
<td>0.49</td>
</tr>
<tr>
<td>( \alpha_2 )</td>
<td>0.075</td>
<td>0.088</td>
<td>0.026</td>
<td>-0.131</td>
<td>0.042</td>
</tr>
<tr>
<td>( \theta )</td>
<td>-0.14</td>
<td>-0.06</td>
<td>0.24</td>
<td>-1.57</td>
<td>0.53</td>
</tr>
<tr>
<td>( \mu )</td>
<td>0.0013</td>
<td>0.0008</td>
<td>0.0006</td>
<td>-0.0005</td>
<td>0.0027</td>
</tr>
<tr>
<td>( \beta )</td>
<td>1.08</td>
<td>1.48</td>
<td>0.24</td>
<td>-0.98</td>
<td>-0.05</td>
</tr>
</tbody>
</table>