UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

ESTIMACIÓN DE PARÁMETROS EN UN MODELO DE VOLATILIDAD ESTOCÁSTICA CON MEMORIA LARGA USANDO FILTRO DE PARTÍCULAS

Tesis sometida a la consideración de la Comisión del Programa de Estudios de Posgrado en Matemática para optar al grado y título de Maestría Académica en Matemática Aplicada.

ANDRÉS QUIRÓS GRANADOS

Ciudad Universitaria Rodrigo Facio, Costa Rica
2021

DEDICATORIA

Dedicado a mi hijo Isaac, mi esposa Jennifer y mi madre Gladys.

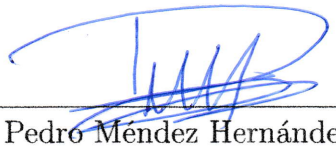
AGRADECIMIENTOS

Agradezco a mi Dios por permitirme este gran logro. *"Porque Jehová da la sabiduría, y de su boca viene el conocimiento y la inteligencia".* Proverbios 2:6.

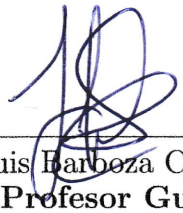
Un sincero agradecimiento a mi tutor Dr. Luis Barboza por su orientación y por sus consejos los cuales ayudaron a cumplir con los objetivos.

Agradezco a mi esposa por todo el apoyo incondicional, por su motivación para poder terminar el proyecto y por su ayuda siendo una lectora más del trabajo.

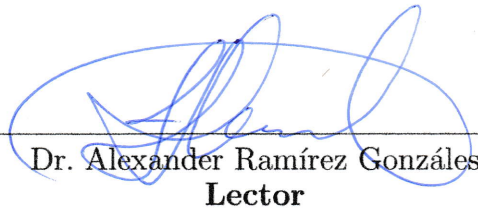
"Esta tesis fue aceptada por la Comisión del Programa de Estudios de Posgrado en Matemática de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Académica en Matemática Aplicada."



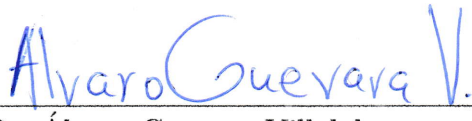
Dr. Pedro Méndez Hernández
Representante del Decano
Sistema de Estudios de Posgrado



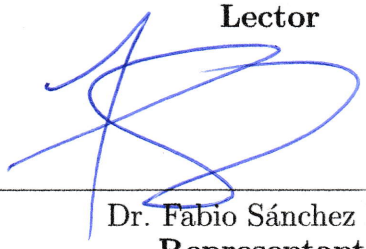
Dr. Luis Barboza Chinchilla
Profesor Guía



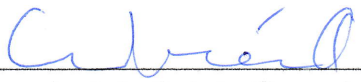
Dr. Alexander Ramírez Gonzáles
Lector



Dr. Álvaro Guevara Villalobos
Lector



Dr. Fabio Sánchez Peña
Representante
Programa de Posgrado en Matemática



Andrés Quirós Granados
Sustentante

TABLA DE CONTENIDOS

RESUMEN

En este estudio se trabaja con un modelo de volatilidad estocástica con memoria larga descrito por medio de un modelo de espacio estado. Se tienen dos procesos estocásticos, uno de ellos cuenta con información observada y el otro es un proceso latente. Las ecuaciones que describen estos procesos cuentan con 1 y 3 parámetros respectivamente. Estos parámetros y las variables del modelo se deben estimar.

El problema se abordó aplicando una combinación del algoritmo de filtro de partículas conocido como filtro de partículas bootstrap y métodos de Cadenas de Marvok vía Monte Carlo. Específicamente se aplicaron los algoritmos: filtro de partículas Liu-West, filtro de partículas marginal Metropolis-Hastings, filtro de partículas secuencial aumentado MCMC y secuencial aumentado MCMC con remuestreo.

Se obtuvo que el filtro de partículas secuencial aumentado MCMC con remuestreo es el mejor algoritmo según las medidas usadas (Raíz del error cuadrático medio y el score de intervalo). Este algoritmo logra estimaciones buenas, tanto para los parámetros como para las variables del modelo. Además se ejecutó el algoritmo con datos reales usando el índice S&P500.

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

UNIVERSIDAD DE COSTA RICA

SEP Sistema de Estudios de Posgrado

Autorización para digitalización y comunicación pública de Trabajos Finales de Graduación del Sistema de Estudios de Posgrado en el Repositorio Institucional de la Universidad de Costa Rica.

Yo, Andrés Quirós Granados, con cédula de identidad 205980124, en mi condición de autor del TFG titulado Estimación de parámetros en un modelo de volatilidad estocástica con memoria larga usando filtro de partículas

Autorizo a la Universidad de Costa Rica para digitalizar y hacer divulgación pública de forma gratuita de dicho TFG a través del Repositorio Institucional u otro medio electrónico, para ser puesto a disposición del público según lo que establezca el Sistema de Estudios de Posgrado. **SI** [X] **NO** *[ ]

***En caso de la negativa favor indicar el tiempo de restricción: _____ año (s).**

Este Trabajo Final de Graduación será publicado en formato PDF, o en el formato que en el momento se establezca, de tal forma que el acceso al mismo sea libre, con el fin de permitir la consulta e impresión, pero no su modificación.

Manifiesto que mi Trabajo Final de Graduación fue debidamente subido al sistema digital Kerwá y su contenido corresponde al documento original que sirvió para la obtención de mi título, y que su información no infringe ni violenta ningún derecho a terceros. El TFG además cuenta con el visto bueno de mi Director (a) de Tesis o Tutor (a) y cumplió con lo establecido en la revisión del Formato por parte del Sistema de Estudios de Posgrado.

INFORMACIÓN DEL ESTUDIANTE:

Nombre Completo: Andrés Quirós Granados.

Número de Carné: A44211 Número de cédula: 205980124.

Correo Electrónico: aquigr@yahoo.es.

Fecha: 15 marzo 2021. Número de teléfono: 8705-1552.

Nombre del Director (a) de Tesis o Tutor (a): Luis Barboza Chinchilla.

FIRMA ESTUDIANTE

Nota: El presente documento constituye una declaración jurada, cuyos alcances aseguran a la Universidad, que su contenido sea tomado como cierto. Su importancia radica en que permite abreviar procedimientos administrativos, y al mismo tiempo genera una responsabilidad legal para que quien declare contrario a la verdad de lo que manifiesta, puede como consecuencia, enfrentar un proceso penal por delito de perjurio, tipificado en el artículo 318 de nuestro Código Penal. Lo anterior implica que el estudiante se vea forzado a realizar su mayor esfuerzo para que no sólo incluya información veraz en la Licencia de Publicación, sino que también realice diligentemente la gestión de subir el documento correcto en la plataforma digital Kerwá.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Antecedentes y justificación

En la matemática financiera se han dado grandes esfuerzos para desarrollar modelos que describan la dinámica de los precios de un activo financiero. Un avance importante fue realizado por Paul Samuelson en los años sesentas al modificar el modelo de Bachelier y así desarrollar el proceso geométrico browniano para el precio de un activo [4], a saber,

$$dP_t = \mu\, P_t\, dt + \sigma\, P_t\, dB_t. \tag{1.1}$$

En la fórmula, el valor μ es la tasa de retorno esperada, σ es la volatilidad del precio del activo y B_t es un movimiento Browniano estándar [18].

El modelo geométrico tiene ventajas sobre el modelo de Bachelier. La expresión (1.1) afirma que el retorno esperado y la volatilidad (o incertidumbre) tienen el mismo valor sin importar el precio inicial del activo, aspecto en lo cual falla en capturar el modelo de Bachelier [18]. Con el modelo de Bachelier los precios pueden tomar valores

negativos con una probabilidad positiva [4]; además los incrementos en los precios siguen una distribución normal con esperanza μ y varianza σ^2, pero hay evidencia con datos observados que las colas son mucho más pesadas que las de la distribución normal [4]. Como el proceso geométrico tiene como solución una función del tiempo exponencial perturbada aleatoriamente [25], supera los dos problemas mencionados. Además es considerado un modelo razonable, como una primera aproximación al precio real de un activo financiero [25].

Se ha observado que la volatilidad varía en el tiempo, por ejemplo se puede ver el capítulo 19 de [18]. En el modelo de Black-scholes [18, 25], el cual asume que el precio del activo subyacente sigue la fórmula (1.1), se obtiene la llamada volatilidad implícita la cual muestra estar relacionada con el precio de ejercicio del activo subyacente. Ésto es una contradicción a lo asumido por el modelo geométrico [6, 7, 18]. Por otra parte, estudios muestran que para algunos activos financieros existe una dependencia de su valor actual con valores pasados [6, 7].

Por los motivos anteriormente mencionados, se ha propuesto que la volatilidad sea descrita por un proceso estocástico y reemplazar la ecuación (1.1) por,

$$dP_t = \mu\, P_t\, dt + \sigma(X_t)\, P_t\, dB_t \tag{1.2}$$

donde $\sigma(x)$ es una función determinística y X_t es un proceso estocástico. En nuestro caso el proceso X_t será un proceso de Ornstein-Uhlenbeck fraccionario [5, 21]. Por lo tanto el modelo de volatilidad estocástica que se va a usar en este estudio es el siguiente:

$$\begin{aligned} dP_t &= \mu\, P_t\, dt + \sigma(X_t)\, P_t\, dB_t \\ dX_t &= \alpha\, X_t\, dt + \beta\, dB_t^H \end{aligned} \tag{1.3}$$

El término B_t^H corresponde a un proceso estocástico browniano fraccionario, si el parámetro $H \in\,]0.5, 1]$ el proceso es de memoria larga. Esto significa que el valor del proceso estocástico está fuertemente correlacionado con valores pasados, es decir la

dependencia en términos de autocorrerlación decae muy lentamente [6]. Por otro lado, se tiene que esta variable aleatoria X_t no tiene observaciones explícitas, es decir es una variable latente.

En la matemática aplicada gran parte de los esfuerzos se enfocan en la estimación de los parámetros de los modelos desarrollados. Buenas aproximaciones de los parámetros producen resultados más confiables y precisos de los modelos. Este proyecto se centra en la estimación conjunta de los parámetros del modelo de volatilidad estocástica con memoria larga (1.3), incluyendo H el parámetro del proceso browniano fraccionario.

En la literatura se encuentran artículos donde se ha trabajado con el modelo (1.3). En el trabajo de Chronopoulou y Viens [6] trataron el problema de obtener el precio de opciones, realizaron un análisis de sensibilidad en la influencia de H sobre los resultados finales. Mostraron un método para estimar el parámetro de memoria larga, de forma separada del resto de parámetros, esto bajo un método implícito en el cual con los datos generaron filtros de partículas para un rango de valores de H para luego escoger el mejor bajo un criterio de error cuadrático medio [6].

En el artículo [7], segunda parte del anterior, Chronopoulou y Viens trabajan el modelo (1.3) con tres versiones, continua, discretización de la versión continua y una versión discreta, pero usan el mismo método para estimar H descrito en [6].

En este trabajo se usó filtro de partículas para abordar el problema de la estimación conjunta de los parámetros de (1.3). Los filtros de partículas son una forma de Monte Carlo secuencial, basado en la técnica de muestreo llamada en inglés *sampling importance resampling* [28].

En su versión original el filtro de partículas, es usado en modelos de espacio-estado[1] para estimar la trayectoria o evolución de la variable latente. Supone que los parámetros del modelo son conocidos, uno de estos algoritmos es el filtro Bootstrap (FB) [12]. Han habido modificaciones en estos algoritmos con el fin de estimar conjuntamente la

[1]El modelo (1.3) es un modelo de espacio-estado. La primera ecuación corresponde a la variable observada y la segunda a la variable latente o no observada.

variable latente y los parámetros, tal es el caso del algoritmo de Liu-West (LW) [24]. El algoritmo de LW introduce un kernel suavisador en la dinámica del muestreo de los parámetros.

Hay otros algoritmos que incorporan los métodos de muestreo de Monte Carlo via cadenas de Markov (MCMC por sus siglas en inglés). Este estudio trabaja con dos de estos algoritmos [19] y [20].

1.2. Objetivos

Los objetivos generales y específicos de este trabajo final de graduación son los siguientes:

1.2.1. Objetivo general

Estimar de forma conjunta los parámetros del modelo de volatilidad estocástica a través de algoritmos de filtros de partículas y MCMC.

1.2.2. Objetivos específicos

- Implementar el algoritmo de Liu-West para estimar de manera conjunta los parámetros del modelo (1.3).
- Implementar algoritmos que combinen filtro de partículas y MCMC.
- Comparar el desempeño de los algoritmos, usando para ello datos simulados y aplicar el mejor algoritmo a una muestra de datos reales.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Proceso estocástico

Partimos de un concepto fundamental dentro de la teoría de la probabilidad, las variables aleatorias. Una variable aleatoria es una función que toma un evento, como puede ser el resultado de un experimento, y le otorga un valor numérico. Esta función es la base para construir herramientas más complejas y sofisticadas que permiten desarrollar modelos matemáticos para describir o pronosticar un fenómeno.

Cuando el tiempo está involucrado en el fenómeno en el que se está interesado se puede utilizar el concepto de proceso estocástico. Se define un proceso estocástico como:

Definición 2.1 ([25]) *Un proceso estocástico X es una colección de variables aleatorias*

$$(X_t,\ t \in T) = (X_t(\omega),\ t \in T,\ \omega \in \Omega),$$

definido en algún espacio Ω.

Una propiedad que es deseable en un proceso, es que su comportamiento en distribución no cambien cuando pasa el tiempo. Matemáticamente esto se define como:

Definición 2.2 ([25]) *Un proceso X es estrictamente estacionario si*

$$(X_{t_1}, \ldots, X_{t_n}) \stackrel{d}{=} (X_{t_1+h}, \ldots, X_{t_n+h}),$$

para todo $t_1, \ldots, t_n \in T$, $n \geq 0$ y h tal que $t_1 + h, \ldots, t_n + h \in T$.

Con la siguiente definición sólo interesa que el comportamiento en distribución de los incrementos del proceso se mantengan sobre el tiempo. Un proceso que tenga esta propiedad no necesariamente es estacionario [25].

Definición 2.3 ([25]) *Un proceso X se dice tener incrementos estacionarios si*

$$X_t - X_s \stackrel{d}{=} X_{t+h} - X_{s+h},$$

para todo $t,\ s \in T$ y h tal que $t + h,\ s + h \in T$.

Además, X se dice tener incrementos independientes si para cualquier escogencia de $t_i \in T$ tal que $t_1 < \cdots < t_n$ y $n \geq 1$,

$$X_{t_2} - X_{t_1}, \ldots, X_{t_n} - X_{t_{n-1}},$$

son independientes.

Una clase importante de procesos estocásticos son los auto-similares. Intuitivamente auto-similaridad significa que, patrones apropiadamente escalados de una trayectoria en cualquier intervalo de tiempo pequeño o grande tiene una forma similar [25].

Definición 2.4 ([2]) *Se dice que X_t es un proceso H-auto-similar, si para algún factor*

c, el proceso rescalado $c^{-H}X_{ct}$, es igual en distribución al proceso original X_t. Es decir,

$$(c^{-H}X_{ct_1}, \ldots, c^{-H}X_{ct_k}) \stackrel{d}{=} (X_{t_1}, \ldots, X_{t_k}).$$

Esta definición es necesaria para los procesos que se usaron en este trabajo.

2.1.1. Proceso estocástico de memoria larga

Existen fenómenos que muestran ciertas características, como lo explica Beran en [2], hay periodos relativamente largos donde las observaciones toman valores altos, pero también hay periodos largos donde las observaciones toman valores bajos. Además, la varianza de la muestra de observaciones parece tener promedio que decae a cero a una tasa más lenta que n^{-1} y la correlación $\hat{\rho}(k)$ decae a cero a una tasa proporcional a $k^{-\alpha}$, con $\alpha \in]0,1[$. Todo esto nos lleva a la siguiente definición:

Definición 2.5 ([2]) *Sea X_t un proceso estacionario. Si existe un $\alpha \in]0,1[$ y una constante $c > 0$ tal que*

$$\lim_{k\to\infty} \frac{\rho(k)}{c\,k^{-\alpha}} = 1.$$

Entonces, X_t es llamado un proceso estacionario con memoria larga.

El parámetro α se puede transformar y cambiar por $H = 1 - \alpha/2$, y el concepto de memoria larga se logra cuando $H \in]0.5, 1[$.

Si un proceso X_t es H-auto-similar con incrementos estacionarios, entonces el proceso $Y_t = X_t - X_{t-1}$ tiene su función de autocorrelación dada por [2],

$$\rho(k) = \frac{1}{2}\left[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right]. \tag{2.1}$$

Usando teoría de series numéricas y desarrollo de Taylor, se llega a que

$$\frac{\rho(k)}{H(2H-1)\,k^{2H-2}} \xrightarrow[k\to\infty]{} 1,$$

y cuando $0.5 < H < 1$ se tiene que la autocorrelación decae muy lentamente a cero, de forma que:

$$\sum_{k=-\infty}^{\infty} \rho(k) = \infty.$$

En resumen, un proceso H-auto-similar con incrementos estacionarios, sus incrementos tienen memoria larga si $H \in]0.5, 1[$.

2.1.2. Movimiento browniano fraccionario y ruido gaussiano fraccionario

Uno de los procesos estocásticos más utilizados es el movimiento browniano estándar, el cual es H-auto-similar con $H = 1/2$. El movimiento browniano estándar es definido como:

Definición 2.6 ([25]) *Un proceso estocástico B es llamado movimiento browniano (estándar) o proceso de Wiener si:*

- $B_0 = 0$,
- B_t *tiene incrementos independientes y estacionarios,*
- *Para todo* $t > 0$, B_t *tiene una distribución normal* $N(0, t)$,
- B_t *tiene trayectoria continua.*

Las trayectorias de un movimiento browniano $B_t(\omega)$, con ω fijo, son extremadamente irregulares a pesar de que hay continuidad. La razón principal de que esto suceda es porque sus incrementos son independientes [25].

En general, el movimiento browniano fraccionario estándar se define como:

Definición 2.7 ([2, 7, 33]) *Un movimiento browniano fraccionario estándar B_t^H es un proceso gaussiano H-auto-similar con $H \in]0, 1[$ e incrementos estacionarios. Su*

distribución es definida por su función de covarianza

$$Cov(B_t^H, B_s^H) = \frac{1}{2}\,[t^{2H} - (t-s)^{2H} + s^{2H}]. \tag{2.2}$$

Ademas, $X_t = B_{k+1}^H$ - B_k^H *es llamado ruido gaussiano fraccionario.*

Note que, si $H = 1/2$, B_t^H es un movimiento browniano estándar [2, 33]. Por otro lado, si $H \in]0.5, 1[$ entonces X_t tiene memoria larga [2].

2.2. Modelo de volatilidad estocástica

Un modelo de volatilidad estocástica es un proceso estocástico en el cual su varianza o volatilidad es aleatoria [18]. Este tipo de modelo fue desarrollado con el fin de superar la debilidad del modelo de Black-Scholes. Al usar el proceso geométrico browniano para describir la dinámica del precio del activo subyacente se asume que la varianza es constante, pero la fórmula de Black-Scholes demuestra una relación entre la volatilidad y el precio de ejercicio del activo subyacente [7].

Lo anterior se puede deducir de las fórmulas para el precio de un contrato call europeo (2.3) y para el precio de un contrato put europeo (2.4) de donde se obtiene la volatilidad implícita [18]:

$$c_t = N(d_1)\,S_t - N(d_2)\,K\,\exp^{-r(T-t)} \tag{2.3}$$

$$p_t = N(-d_2)\,K\,\exp^{-r(T-t)} - N(-d_1)\,S_t \tag{2.4}$$

donde,

$$d_1 = \frac{\ln(S_t/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.5}$$

$$d_2 = d_1 - \sigma\sqrt{T-t} \tag{2.6}$$

y N es la función de distribución de una normal estándar, $(T-t)$ es el tiempo al vencimiento, S_t es el precio spot, K es el precio de ejercicio, r es la tasa libre de riesgo y σ es la volatilidad del retorno del activo subyacente.

También hay evidencia documentada de la llamada persistencia de la volatilidad [9, 6, 33]. La persistencia de la volatilidad significa que la volatilidad de hoy nos dice algo no sólo sobre la volatilidad de mañana, sino también sobre la volatilidad de más días en el futuro. En este sentido, Comte [9] introduce el concepto de memoria larga en el modelo continuo de volatilidad estocástica de Hull and White.

En este estudio se trabajó con el modelo propuesto por Comte [9]. En este modelo la volatilidad se describe a través de un proceso estocástico, con el fin de solucionar el inconveniente descrito anteriormente.

$$dP_t = \mu\, P_t\, dt + \sigma(X_t)\, P_t\, dB_t \tag{2.7}$$

$$dX_t = -\alpha\, X_t\, dt + \beta\, dB_t^H \tag{2.8}$$

Pero en vez de usar directamente el precio se usa el logaritmo del precio $Y_t = \log(P_t)$, con lo cual el modelo es el siguiente:

$$dY_t = \left(\mu - \frac{\sigma(X_t)^2}{2}\right) dt + \sigma(X_t)\, dB_t \tag{2.9}$$

$$dX_t = -\alpha\, X_t\, dt + \beta\, dB_t^H \tag{2.10}$$

La ecuación (2.9) describe la dinámica del precio de un activo financiero, es un proceso geométrico browniano con la volatilidad definida por $\sigma(x)$ una función determinística y X_t un proceso estocástico. La dinámica de Y_t se describe con la ecuación (2.10), un proceso de Ornstein-Uhlenbeck fraccionario [5, 21]. En los artículos de [6] y [7] se utiliza este modelo.

2.3. Modelo espacio estado

Un modelo de espacio estado describe la dependencia probabilística entre dos variables aleatorias a través del tiempo y para ello involucra dos series de tiempo. Una de las series (y_t) cuenta con observaciones y la otra serie (x_t) no tiene observaciones, en otras palabras es una serie de tiempo de variables latentes. El modelo define la relación entre las series de la siguiente forma [22]:

$$x_t = F_t(x_{t-1}, \epsilon_t) \tag{2.11}$$

$$y_t = H_t(x_t, \varepsilon_t) \tag{2.12}$$

donde ϵ_t y ε_t son errores no necesariamente gaussianos, F y G son funciones no lineales.

La primer ecuación (2.11) llamada ecuación de estado, relaciona el estado actual del sistema con el estado anterior tomando en cuenta un componente de error. La ecuación (2.12) se le llama ecuación de observación y relaciona las observaciones con el estado actual del modelo agregando un error [4].

En los modelos de espacio estado el principal problema es la estimación del vector de estados $x_{0:t} = \{x_1, \ldots, x_t\}$ a partir del vector de observaciones $y_{0:t} = \{y_1, \ldots, y_t\}$ [22]. Lo anterior se puede hacer en un sentido bayesiano, estimando la distribución posterior de todos los estados dada todas las observaciones, esto es

$$p(x_{0:t} \,|\, y_{1:t}) = \frac{p(y_{1:t} \,|\, x_{0:t})\, p(x_{0:t})}{p(y_{1:t})}.$$

Cada vez que se obtiene una nueva observación se tiene que recalcular la distribución y cuando t es grande el cálculo prácticamente no se puede realizar [12, 32]. En lugar de estimar la distribución completa se puede trabajar con la distribución marginal del estado latente y con un modelo probabilistico con propiedades de Markov [12, 23, 32], así el modelo de espacio estado descrito por las ecuaciones (2.11) y (2.12) se puede

reescribir como,

Probabilidad inicial	$x_0 \sim p(x_0)$
Probabilidad de transición	$x_k \sim p(x_t \mid x_{t-1}), \quad t \geq 1$
Probabilidad de la observación dado el estado	$y_t \sim p(y_t \mid x_t), \quad t \geq 1.$

El modelo va a tener las siguientes propiedades.

Propiedad 2.1 ([32]) *El estado actual x_t dado x_{t-1} es condicionalmente independiente de los estados pasados antes del tiempo $t-1$, esto es,*

$$p(x_t \mid x_{1:t-1}, y_{1:t-1}) = p(x_t \mid x_{t-1}).$$

Propiedad 2.2 ([32]) *La observación actual y_t dado x_t es condicionalmente independiente de los estados y observaciones anteriores, esto es,*

$$p(y_t \mid x_{1:t}, y_{1:t-1}) = p(y_t \mid x_t).$$

Las distribuciones que se van a considerar ahora son [22]:

Tipo	Fórmula
Filtro	$p(x_t \mid y_{1:t})$
Predicción	$p(x_{t+n} \mid y_{1:t})$

y usando la fórmula de Bayes y la ecuación de Chapman-Kolmogorov se obtienen las siguientes fórmulas [12, 23] para obtener las distribuciones,

$$p(x_t \mid y_{1:t-1}) = \int p(x_t|x_{t-1})\, p(x_{t-1} \mid y_{1:t-1})\, dx_{t-1} \tag{2.13}$$

$$p(x_t \mid y_{1:t}) = \frac{p(y_t \mid x_t)\, p(x_t \mid y_{1:t-1})}{\displaystyle\int p(y_t \mid x_t)\, p(x_t \mid y_{1:t-1})\, dx_t} \tag{2.14}$$

2.4. Filtro de partículas

Una de las formas de obtener una estimación de las distribuciones (2.13) (2.14) es por medio de simulación de Monte Carlo. La forma directa de aplicar Monte Carlo es obteniendo muestras de las distribuciones involucradas [17, 29, 30], en nuestro caso $p(x \mid y_{1:t})$, pero no siempre es posible.

En términos generales se quiere ser capaz de estimar

$$E[g(x) \mid y_{1:t}] = \int g(x)\, p(x \mid y_{1:t})\, dx.$$

Otra alternativa para realizar la estimación es usando el método *Muestreo por importancia* [30, 31]. Se incluye una distribución llamada distribución de muestreo por importancia $\pi(x \mid y_{1:t})$ de la cual se puede obtener muestras fácilmente [12]. El método está basado en la siguiente descomposición, en donde se asume que el soporte de π contiene el soporte de $p(x \mid y_{1:t})$ [30],

$$\begin{aligned} E[g(x) \mid y_{1:t}] &= \int g(x)\, p(x \mid y_{1:t})\, dx && (2.15) \\ &= \int g(x) \left[\frac{p(x \mid y_{1:t})}{\pi(x \mid y_{1:t})}\right] \pi(x \mid y_{1:t})\, dx && (2.16) \\ &= \int g(x)\, \omega(x)\, \pi(x \mid y_{1:t})\, dx. && (2.17) \end{aligned}$$

Para aproximar (2.17) se utiliza simulación Monte Carlo, al obtener muestras de la distribución de muestreo por importancia:

$$x^{(i)} \sim \pi(x \mid y_{1:t}), \quad i = 1, \ldots, N$$

y la aproximación sigue de la forma [12, 32]:

$$E[g(x) \mid y_{1:t}] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(x^{(i)} \mid y_{1:t})}{\pi(x^{(i)} \mid y_{1:t})} g(x^{(i)}) \tag{2.18}$$

$$= \sum_{i=1}^{N} \tilde{\omega}(x^{(i)})\, g(x^{(i)}). \tag{2.19}$$

La densidad posterior se estima con este método como [12],

$$p(x \mid y_{1:t}) \approx \sum_{i=1}^{N} \tilde{\omega}(x^{(i)})\, \delta_x(x^{(i)}), \tag{2.20}$$

donde $\delta(x)$ es la función delta de Dirac.

Como se menciona en [12] este método no es adecuado cuando se tiene que calcular la evolución secuencial de distribuciones posteriores, cada vez que una observación es disponible se tiene que recalcular los pesos y con el paso del tiempo se complican más los cálculos.

Para poder sobrellevar el inconveniente anteriormente mencionado se desarrolla una versión llamada *Muestreo secuencial por importancia* [12, 22, 30, 32]. La modificación parte de la distribución completa posterior y de las propiedades 2.1 y 2.2 para tener una fórmula recursiva [12]:

$$p(x_{0:t} \mid y_{1:t}) = p(x_{0:t-1} \mid y_{1:t-1}) \frac{p(y_t \mid x_t)\, p(x_t \mid x_{t-1})}{p(y_t \mid y_{1:t-1})} \tag{2.21}$$

$$\propto p(x_{0:t-1} \mid y_{1:t-1})\, p(y_t \mid x_t)\, p(x_t \mid x_{t-1}). \tag{2.22}$$

Si se escoge la distribución de muestreo por importancia tal que cumpla:

$$\pi(x_{0:t} \mid y_{1:t}) = \pi(x_t \mid x_{0:t-1}, y_{1:t})\, \pi(x_{0:t-1} \mid y_{1:t-1}),$$

se tiene que los pesos también se pueden calcular de forma recursiva,

$$\omega_t^{(i)} \propto \frac{p(x_{0:t-1}^{(i)} \mid y_{1:t-1})}{\pi(x_{0:t-1}^{(i)} \mid y_{1:t-1})} \frac{p(y_t \mid x_t^{(i)})\, p(x_t^{(i)} \mid x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{1:t})} \tag{2.23}$$

$$= \omega_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)})\, p(x_t^{(i)} \mid x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{1:t})}. \tag{2.24}$$

De esta forma sólo hay que obtener muestras de x_t e ir construyendo $x_{0:t} = \{x_{0:t-1}, x_t\}$.

Algoritmo 2.1 Remuestreo secuencial por importancia o Filtro de partículas

1: Obtener muestras $x_0^{(i)}$ de la distribución a priori y tomar $\omega_0^{(i)} = 1/N$,

$$x_0^{(i)} \sim p(x_0), \quad i = 1, \ldots, N.$$

2: **Para** $t = 1, \ldots, T$ **hacer**

3: Obtener muestras $x_t^{(i)}$ de la distribución por importancia

$$x_t^{(i)} \sim \pi(x_t \mid x_{t-1}^{(i)}, y_{1:t}), \quad i = 1, \ldots, N.$$

4: Calcular los pesos nuevos

$$\omega_t^{(i)} \propto \frac{p(y_t \mid x_t^{(i)})\, p(x_t^{(i)} \mid x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid x_{t-1}^{(i)}, y_{1:t})},$$

y normalizarlos para que sumen 1.

5: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_t^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{\omega_t^{(i)} : i = 1, \ldots, N\}$, luego tomar $\omega_t^{(i)} = 1/N$.

6: **Fin Para**

El muestreo secuencial por importancia también tiene sus debilidades, hay que ir almacenando la historia de los estados $x_{0:t}$ para poder aplicar el algoritmo [32]. Esto se resuelve escogiendo la distribución de muestreo por importancia de forma conveniente [32],

$$\pi(x_t \mid x_{0:t-1}, y_{1:t}) = \pi(x_t \mid x_{t-1}, y_{1:t}).$$

Además se da el problema de degeneración [30], fácilmente se puede encontrar la situación de que después de un tiempo casi todas las partículas tienen pesos con valor muy cercano a cero. Para solucionar el problema de la degeneración se incorpora un remuestreo con el fin de eliminar las partículas con pesos muy pequeños y duplicar las partículas con pesos grandes, esto da lugar a la versión de muestreo por importancia llamado *Remuestreo secuencial por importancia* también conocido como *Filtro de partículas* [12, 22]. El Algoritmo 2.1 resume el método.

Algoritmo 2.2 Filtro de partículas bootstrap

1: Obtener muestras $x_0^{(i)}$ de la distribución a priori y tomar $\omega_0^{(i)} = 1/N$,

$$x_0^{(i)} \sim p(x_0), \quad i = 1, \ldots, N.$$

2: **Para** $t = 1, \ldots, T$ **hacer**
3: Obtener muestras $x_t^{(i)}$ de la distribución por importancia

$$x_t^{(i)} \sim p(x_t \,|\, x_{t-1}^{(i)}), \quad i = 1, \ldots, N.$$

4: Calcular los pesos nuevos

$$\omega_t^{(i)} \propto p(y_t \,|\, x_t^{(i)}),$$

y normalizarlos para que sumen 1.
5: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_t^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{\omega_t^{(i)} : i = 1, \ldots, N\}$, luego tomar $\omega_t^{(i)} = 1/N$.
6: **Fin Para**

Dentro de las variaciones al algoritmo de filtro de partículas la más conocida es el *Filtro de partículas bootstrap*. Usa como distribución de muestreo por importancia la distribución de transición $p(x_t \,|\, x_{t-1})$ del modelo, con esto los pesos se obtienen fácilmente, $\omega_t^{(i)} \propto p(y_t \,|\, x_t^{(i)})$. La implementación puede ser muy fácil pero puede requerir un gran número de partículas para tener buenos resultados [32]. El Algoritmo 2.2 resume el método.

Otra variación importante, es el llamado *Filtro de partículas auxiliar* [26]. Realiza un

remuestreo de las partículas del paso anterior con pesos proporcionales a $p(y_t \mid g(x_{t-1}^{(i)}))$, donde g puede ser la esperanza o moda de $p(x_t \mid x_{t-1})$. La idea es tener las partículas con mayor probabilidad de que su propagación esté en la zona de la densidad posterior. El Algoritmo 2.3 resume el método.

Algoritmo 2.3 Filtro de partículas auxiliar

1: Obtener muestras $x_0^{(i)} \sim p(x_0)$ de la distribución a priori y tomar $\omega_0^{(i)} = 1/N, \quad i = 1, \ldots, N$.
2: **Para** $t = 1, \ldots, T$ **hacer**
3: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_{t-1}^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{w_t^{(i)} = p(y_t \mid g(x_{t-1}^{(i)})) : i = 1, \ldots, N\}$.
4: Obtener muestras $x_t^{(i)}$ de la distribución por importancia,

$$x_t^{(i)} \sim p(x_t \mid \tilde{x}_{t-1}^{(i)}), \quad i = 1, \ldots, N.$$

5: Calcular los pesos nuevos

$$\omega_t^{(i)} \propto \frac{p(y_t \mid x_t^{(i)})}{p(y_t \mid g(\tilde{x}_{t-1}^{(i)}))},$$

y normalizarlos para que sumen 1.
6: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_t^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{\omega_t^{(i)} : i = 1, \ldots, N\}$, luego tomar $\omega_t^{(i)} = 1/N$.
7: **Fin Para**

2.4.1. Estimación de parámetros

El modelo de espacio estado (2.11), (2.12) por lo general tiene parámetros $\boldsymbol{\theta}$ no conocidos que se deben estimar, así incorporar los parámetros permite que el modelo

pueda ser ajustado a datos reales [32]:

Probabilidad inicial de los parámetros	$\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
Probabilidad inicial del sistema	$x_0 \sim p(x_0 \mid \boldsymbol{\theta})$
Probabilidad de transición	$x_k \sim p(x_t \mid x_{t-1}, \boldsymbol{\theta}), \quad t \geq 1$
Probabilidad de la observación dado el estado	$y_t \sim p(y_t \mid x_t, \boldsymbol{\theta}), \quad t \geq 1.$

Y la distribución que estamos interesados en estimar es:

$$p(x_t, \boldsymbol{\theta} \mid y_{1:t}) \propto p(y_t \mid x_t, \boldsymbol{\theta})\, p(x_t \mid y_{1:t-1}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid y_{1:t-1}).$$

2.5. Cadenas de Markov vía Monte Carlo

Las cadenas de Markov vía Monter Carlo (MCMC por sus siglas en inglés) son métodos que generan muestras para $\boldsymbol{\theta}$ provenientes de la distribución posterior $p(\boldsymbol{\theta} \mid y)$ cuando esta distribución no puede ser directamente simulada [29].

Los métodos MCMC son Monte Carlo porque dependen de muestreos aleatorios. Producen cadenas de Markov pues las muestras que se generan no son independientes, cada muestra depende de la anterior. Además su distribución estacionaria es $p(\boldsymbol{\theta} \mid y)$ [30].

Un mecanismo para la generación de una cadena de este tipo es llamado algoritmo de Metropolis-Hastings el cual se puede describir de la siguiente manera:

1. Generar $\boldsymbol{\theta}^* \sim J(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(s)})$

2. Calcular la tasa de aceptación

$$\alpha = \frac{p(\boldsymbol{\theta}^* \mid y)}{p(\boldsymbol{\theta}^{(s)} \mid y)} = \frac{p(y \mid \boldsymbol{\theta}^*)\, p(\boldsymbol{\theta}^*)}{p(y \mid \boldsymbol{\theta}^{(s)})\, p(\boldsymbol{\theta}^{(s)})}$$

3. Hacer

$$\boldsymbol{\theta}^{(s+1)} = \begin{cases} \boldsymbol{\theta}^* & \text{con probabilidad } \min(\alpha, 1) \\ \boldsymbol{\theta}^{(s)} & \text{con probabilidad } 1 - \min(\alpha, 1). \end{cases}$$

La distribución $p(\boldsymbol{\theta} \mid y)$ es llamada la distribución objetivo y $J(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(s)})$ es llamada la distribución propuesta.

La propuesta más común de usar (desde que Hastings propuso su uso [30]) es la caminata aleatoria

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(s)} + \epsilon$$

y las distribuciones más usadas para la perturbación ϵ suelen ser uniforme, normal o Cauchy [29].

En la práctica se suele no considerar las primeras B iteraciones del proceso, es decir sólo se considera la muestra $\{\boldsymbol{\theta}^{B+1}, \ldots, \boldsymbol{\theta}^m\}$. Estas B iteraciones iniciales se le llama periodo de quema y el propósito es no considerar el periodo en que la cadena se mueve desde su valor inicial hasta la zona del espacio de parámetros con mayor probabilidad posterior [17].

2.6. Estado actual del problema

El problema que se trata en este estudio tiene sus antecedentes en los artículos de Chronopoulou [6, 7], en el cual el problema de interés fue estimar el precio de opciones considerando el modelo (2.9), (2.10) para el precio de un activo subyacente. Parte del artículo [6] describió como estimar los parámetros del modelo incluyendo el parámetro de memoria larga. La estimación no se realizó de forma conjunta, se usó un método para estimar H para luego estimar los otros parámetros del modelo (2.9), (2.10). La forma de obtener H es la siguiente: toman valores de H desde 0.5 hasta 0.95 con incrementos de 0.01, para cada uno de los valores estiman la distribución de la volatilidad, calculan el

correspondiente precio de la opción para varios precios de ejercicio, para luego obtener el error cuadrático medio del precio de la opción con respecto al bid-ask spread. Se escoge el H correspondiente al valor más pequeño del error cuadrático medio.

Con respecto al método usado en este trabajo, filtro de partículas, se tiene conocimiento de varios artículos con información valiosa. Recientemente se ha incorporado en los filtros de partículas las cópulas con el fin de estimar los parámetros de un modelo de espacio estado. El artículo de Deng [10] desarrolla un modelo de espacio estado en el cual la varianza del error es función del estado del sistema, además modifica el filtro Liu-West incorporandole cópulas para considerar la dependencia de los parámetros con la variable del sistema y las observaciones. Se cambia la forma de calcular los pesos, se usa la función de densidad de la cópula para calcularlos:

$$w_t^{(i)} = c(y_t \mid g(x_{t-1}^{(i)}), m_{t-1}^{(i)})$$

$$\omega_t^{(i)} \propto \frac{c(y_t \mid x_t^{(i)}, \boldsymbol{\theta}_t^{(i)})}{c(y_t \mid g(\tilde{x}_{t-1}^{(i)}), \tilde{m}_{t-1}^{(i)})},$$

donde c es la función de densidad de la cópula. El artículo concluye, basado en los resultados numéricos del error cuadrático medio, que el filtro de partículas propuesto es superior al filtro de Liu-West.

El artículo de Fan [13] desarrolla un complicado filtro de partículas incluyendo cópulas, el cual lo aplica a problemas de hidrología. El trabajo con cópulas, además de ser usadas para incluir las estructuras de dependencia, se hace para corregir dos problemas que se dan con los filtros de partículas, a saber, la degeneración de las partículas y el empobrecimiento de las muestras. Se concluye que el método propuesto tiene más exactitud que los filtros tradicionales usando muestras de tamaño de 100 partículas y que el tamaño de la muestra de partículas no afecta significativamente el desempeño del algoritmo propuesto.

Un artículo importante es el de Jacob [19], en donde se trabajó en el campo de

la electroquímica. Usan un modelo de espacio estado donde la variable latente no es Markov. Los autores indican que los métodos de filtros y filtros MCMC se pueden implementar directamente en procesos que no cumplen la propiedad de Markov. El problema de la estimación de parámetros se aborda con un método de cadenas de Markov vía Monte Carlo[1] llamado Filtro de partículas marginal Metropolis-Hastings. El artículo estudia la identificación de los parámetros y su sensibilidad a la escogencia de distribuciones a priori y al número de observaciones. Con respecto a la inferencia bayesiana se sugiere probar con otros métodos como Filtro de partículas Gibbs y SMC^2.

Otro artículo importante para este estudio es el de Javvad [20]. En el artículo se implementa un algoritmo denominado secuencial aumentado de cadenas de Markov vía Monte Carlo para obtener la distribución posterior de los parámetros. El algoritmo se desarrolla mejorando los métodos secuenciales actuales en los que se debe contar con estadísticos suficientes o el uso de un modelo artificial para la evolución de los parámetros.

[1]MCMC por sus siglas en inglés.

CAPÍTULO 3

MÉTODO DE TRABAJO

El estudio se llevará a cabo con dos tipos de datos, datos simulados y datos reales. Los datos simulados nos permiten identificar debilidades y beneficios de cada uno de los algoritmos, así como realizar comparaciones entre los diferentes métodos, mientras que con los datos reales se puede confirmar la aplicabilidad de los filtros de partículas para un problema en específico [28].

Los algoritmos se implementaron en el lenguaje estadístico R [27]. Por otro lado, se usa el promedio como estimador de los parámetros y variables objetivos del estudio.

Este capítulo es organizado de la siguiente forma: en la sección 3.1 se indica como se obtuvieron los datos simulados, además de las medidas usadas para medir el desempeño de los algoritmos. La sección 3.2 indica cuales datos reales fueron usados. La implementación de los algoritmos se trata en la sección 3.3, y finalmente en la sección 3.4 se detallan los algoritmos que se usaron para resolver el problema.

3.1. Análisis de los datos simulados

3.1.1. Datos simulados

Para medir el rendimiento de los algoritmos se utilizaron datos simulados de mediciones diarias en un periodo de tiempo, utilizando el modelo (2.9), (2.10). Para esto se usó la discretización de Euler de primer orden [7], a saber,

$$Y_{t+1} = Y_t + (\mu - X_t^2/2)\,dt + X_t\,\epsilon_t\,\sqrt{dt} \tag{3.1}$$

$$X_{t+1} = X_t - \alpha\,X_t\,dt + \beta\,(B_{t+1}^H - B_t^H). \tag{3.2}$$

Donde, $Y_t = \log(P_t)$ es el logaritmo del precio del stock, X_t es la volatilidad, ϵ_t es el error que sigue una distribución normal estándar, $dt = 1/N$ y B_t^H es un movimiento browniano fraccionario estándar.

Se tomaron valores específicos para los parámetros, a saber: $\alpha = 0.02733$, $\beta = 0.07567$, $\mu = 0.0014$, $H = 0.6$, $x_0 = 6.802$, $y_0 = 0.35$ y se tomó $\sigma(x) = x$, función que se usa en la ecuación (2.9). A excepción de μ, estos valores se usaron en [6]. Además se definieron valores para N (el número de partículas) y T (cantidad de datos históricos), de 130 y 255 respectivamente. Las series simuladas de las variables se pueden observar en la Figura 3.1.

En términos generales se realizaron pruebas de sensibilidad en los algoritmos, con el fin de observar el efecto en los resultados. En el algoritmo LW se varió el número de partículas. En PMMH se cambió la distribución a priori con el objetivo de ver la influencia en los resultados, se varió de una distribución normal multivariada a una gamma multivariada. Para el algoritmo SAMCMC se varió el periodo de quema en los MCMC's y el número de partículas. Por otro lado, se realizaron mediciones del desempeño tales como: precisión, capacidad para predecir y tiempo de ejecución de los diferentes algoritmos. Con estas medidas se determinó cual de los algoritmos es mejor.

Los algoritmos se ejecutaron diez veces ($K = 10$), con el objetivo de aislar el efecto de la aleatoriedad de la simulación en las medidas de desempeño.

Figura 3.1: *Caminos simulados de las variables Log-Precio y volatilidad usados para ejecutar los algoritmos.*

3.1.2. Precisión de la estimación

Para determinar la precisión de la estimación, esto es medir la desviación entre el valor real y el valor estimado, se calculó para cada K la raíz del error cuadrático medio (RECM) y el valor que se reporta es le promedio sobre K [13, 20, 28].

La medida se calcula para las variables del modelo espacio estado X_t y Y_t, y también

para los parámetros del modelo α, β, H y μ. Recordemos que lo que se busca en la simulación es analizar la capacidad predictiva de los estimadores en parámetros y en la variable latente.

$$\overline{RECM} = \frac{1}{K}\sum_{k=1}^{K} RECM_k = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{T}\sum_{t=1}^{T}\sqrt{\sum_{j=1}^{N}\frac{(x_t - \hat{X}_{j,t,k})^2}{N}}. \tag{3.3}$$

$$\overline{RECM} = \frac{1}{K}\sum_{k=1}^{K} RECM_k = \frac{1}{K}\sum_{k=1}^{K}\sqrt{\sum_{j=1}^{N}\frac{(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{j,k})^2}{N}}. \tag{3.4}$$

Donde, x_t es el valor observado simulado de las variables del modelo espacio estado $\hat{X}_{j,t,k}$ es la muestra j del valor estimado para el momento t en la ejecución k y T es largo del camino observado. $\boldsymbol{\theta}$ es el valor del parámetro, $\hat{\boldsymbol{\theta}}_{j,k}$ es el valor estimado y N es el tamaño de la muestra.

Un valor de $\overline{RECM}$ cercano a cero refleja en forma total la cercanía del estimador al valor real e indica que el algoritmo en estudio es efectivo en inferir la variable en el modelo [20].

3.1.3. Capacidad para predecir

El score de intervalo [14] es un indicador diseñado para predicciones en formato de intervalos de percentiles, este indicador sólo requiere del intervalo de predicción central $(1-\alpha)\times 100\,\%$. La definición del score es:

$$IS_{1-\alpha} = (u-l) + \frac{2}{\alpha}(l-x)\,\mathbb{1}_{\{x<l\}} + \frac{2}{\alpha}(x-u)\,\mathbb{1}_{\{x>u\}}. \tag{3.5}$$

Donde $\mathbb{1}$ es la función indicadora, l es el percentil $\alpha/2$ y u es el percentil $1-\alpha/2$ de la distribución empírica posterior de la variable del modelo espacio estado o del parámetro.

El objetivo es medir la capacidad predictiva del estimador, al recompensar los inter-

valos de predicción más angostos y penalizar si el valor real queda afuera del intervalo de predicción central. Note que entre más ancho el intervalo de predicción central más severa es la penalización [3].

Se reporta para la estimación de los parámetros el promedio de los score de intervalo de las K ejecuciones.

$$\overline{IS}_{1-\alpha} = \frac{1}{K}\sum_{k=1}^{K} IS_{1-\alpha,k} \tag{3.6}$$

Para la estimación de las variables del modelo de espacio estado se usa la siguiente fórmula:

$$\overline{IS}_{1-\alpha} = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=1}^{T} \frac{IS_{1-\alpha,t,k}}{T} \tag{3.7}$$

donde T es el tamaño del del camino.

Se calculó la medida con valores de α iguales a 0.1 y 0.2.

3.1.4. Tiempo computacional

El tiempo de ejecución de los algoritmos se midió en segundos. Se reporta el promedio de las K ejecuciones. Si la duración es muy larga el resultado se muestra en horas.

3.2. Análisis de los datos reales

Para el segundo tipo de datos se usó información real, específicamente el índice Standard & Poor's 500 (S&P500) [34], uno de los más importantes de Estados Unidos. El índice será usado en la ejecución del algoritmo con mejor desempeño.

Se tomó una serie corta y una larga del índice. Para la serie corta se tomaron datos entre las fechas 2008-12-29 al 2009-01-29, para un total de 22 observaciones. Para la serie larga se tomaron 252 observaciones del periodo de 2008-12-29 al 2009-12-28.

Los datos se tomaron de Yahoo Finanzas y se presentan en el anexo A.2.

3.3. Implementación

En esta sección se detallan los temas involucrados en la implementación de los métodos.

3.3.1. Software y Hardware

Los algoritmos se ejecutaron en una computadora con las siguientes características: arquitectura x86_64, procesador Inter (R) Xeon (R) CPU E5-2630 v3 2.40GHZ, 16 núcleos, 62 Gb memoria RAM.

Se usó el lenguaje estadístico R [27] para programar y ejecutar los algoritmos en un sistema Linux Ubuntu 18.04.5 LTS. La versión de R que se utilizó fue 3.6.3.

3.3.2. Obtención del número de partículas

Para determinar un número de partículas se siguió el procedimiento descrito en [19]. La estrategia consiste en tomar una muestra de la distribución a priori de $\boldsymbol{\theta}$, se ejecuta el filtro de partículas 100 veces con el mismo $\boldsymbol{\theta}$. Se genera una serie de valores $Z_1, \ldots, Z_{100}$ de $p(y_{0:T} \mid \boldsymbol{\theta})$. Luego, se imita un Metropolis-Hastings usando como probabilidad $\alpha = \text{mín}(1, Z_j/Z_{\text{actual}})$ para aceptar $Z_{\text{actual}} = Z_j$. Al final, se obtiene la tasa de veces que un nuevo Z fue aceptado.

Según [19] una tasa de 10 % se puede considerar buena. Este procedimiento se ejecutó con varios valores de $\boldsymbol{\theta}_0$ y número de partículas. Los resultados se pueden ver en el cuadro 3.1.

Se tomó la decisión de usar 130 partículas como escenario base debido a que con las pruebas que se realizaron con 130 se tiene más de 10 % de tasa de aceptación.

Cuadro 3.1: *Resultados de procedimiento para determinar el número de partículas.*

$\boldsymbol{\theta}_0 = (\alpha_0, \beta_0, H_0, \mu_0)$	Partículas	Tasa
(0.06715, 0.05978, 0.78726, 0.00768)	50	15 %
(0.05298, 0.06798, 0.26147, 0.00467)	130	11 %
(0.01295, 0.07005, 0.41802, 0.00505)	100	11 %
(0.02344, 0.05909, 0.60494, 0.00700)	100	10 %
(0.05361, 0.04691, 0.97870, 0.00157)	130	20 %

3.4. Algoritmos

En esta sección se describe los algoritmos que se implementaron, en particular: filtro de partículas Liu-West, filtro de partículas marginal Metropolis-Hastings, filtro de partículas secuencial aumentado MCMC y filtro de partículas secuencial aumentado MCMC con remuestreo.

Además de estos algoritmos, se indica el algoritmo usado para la generación de los caminos del movimiento browniano fraccionario estándar.

3.4.1. Movimiento browniano fraccionario estándar

Para generar caminos del movimiento browniano fraccionario estándar B^H, se usa la descomposición de Choleski para la matriz de covarianza (2.2),

$$(\Gamma)_{i,j} = \operatorname{cov}\left(\frac{i}{N}, \frac{j}{N}\right), \quad \text{para } i, j = 0, \ldots, N-1. \tag{3.8}$$

Donde Γ es simétrica definida positiva y admite la descomposición de Choleski, $\Gamma = LL^t$ con L matriz triangular inferior.

Para simular un camino de B^H, se genera un vector Z de una distribución normal estándar multivariada, se calcula LZ y la estimación del camino es $\tilde{B}^H = (0, (LZ)^t)^t$

[8] (al igual que en la definición 2.6 se asume que el proceso inicia en 0).

Este método es exacto, pero tiene una complejidad computacional de $O(N^3)$ [8].

3.4.2. Filtro de partículas Liu-West

Una técnica para poder incorporar la estimación de parámetros en los Algoritmos 2.2 y 2.3 es la llamada *Kernel suavisador* (KS), en donde se asume que la distribución de $\boldsymbol{\theta}$ es una mezcla de normales multivariadas [24].

Si se tiene un conjunto de partículas $\{x_{t-1}^{(i)}, \boldsymbol{\theta}_{t-1}^{(i)} : i = 1, \ldots, N\}$ con sus respectivos pesos $\{\omega_{t-1}^{(i)} : i = 1, \ldots, N\}$ que aproximan a $p(x_{t-1}, \boldsymbol{\theta} \,|\, y_{1:t-1})$, se tendría que la distribución posterior de $\boldsymbol{\theta}$ se puede aproximar de la siguiente forma:

$$p(\boldsymbol{\theta} \;\; | \, y_{1:t-1}) \approx \sum_{i=1}^{N} \omega_{t-1}^{(i)} \, N(\boldsymbol{\theta} \,|\, m_{t-1}^{(i)}, h^2 V_{t-1}),$$

donde,

$$m_{t-1}^{(i)} = a\, \boldsymbol{\theta}_{t-1}^{(i)} + (1-a)\, \overline{\boldsymbol{\theta}},$$

$$\overline{\boldsymbol{\theta}} = \sum_{i=1}^{N} \frac{\boldsymbol{\theta}_{t-1}^{(i)}}{N},$$

$$V_{t-1} = \sum_{i=1}^{N} \frac{[\boldsymbol{\theta}_{t-1}^{(i)} - \overline{\boldsymbol{\theta}}][\boldsymbol{\theta}_{t-1}^{(i)} - \overline{\boldsymbol{\theta}}]'}{N},$$

$N(\boldsymbol{\theta} \,|\, m, V)$ es una distribución normal multivariada, con esperanza m y matriz de covarianzas V. Además la constante $a^2 = 1 - h^2$ es un factor de contracción [24].

Combinando el Filtro de partículas auxiliar con kernel suavisador se obtiene el Filtro de Liu and West (LW) [23, 24, 28]. El Algoritmo 3.1 resume el método.

Para aplicar este algoritmo se usó un factor de contracción, recomendado en [24] de $a = 0.99495$. La función $g(x)$ que se usó es la esperanza. Los valores iniciales $\boldsymbol{x}_0$, $\boldsymbol{y}_0$

y $\boldsymbol{\theta}_0$ se obtienen de distribuciones normales multivariadas:

$$\boldsymbol{x}_0 \sim N(\mu_x, \sigma_x, \text{mín} = 0, \text{máx} = 1), \tag{3.9}$$

donde, $\mu_x = \text{sd}(\nabla y^{obs})/\sqrt{1/N}$, y $\sigma_x = 1$.

$$\boldsymbol{y}_0 \sim N(\mu_y, \sigma_y, \text{mín} = 0, \text{máx} = \infty) \tag{3.10}$$

donde, $\mu_y = \overline{y}^{obs}$, $\sigma_y = \text{var}(y^{obs})$.

$$\boldsymbol{\theta}_0 \sim N(\mu_\theta, \Sigma_\theta, \text{mín} = (0,0,0,0), \text{máx} = (1,1,1,1)), \tag{3.11}$$

donde, $\mu_\theta = (0, 0, 0.75, 0)$, $\Sigma_\theta = \text{diag}(5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 5 \cdot 10^{-2}, 15 \cdot 10^{-7})$. La escogencia de estos valores no afecta la calidad del ajuste del filtro de Liu-West.

Para obtener los pesos w se utilizó la medida de probabilidad discreta usada en [6], la cual tiene la fórmula:

$$p(y_t \mid x_t^{(i)}, \boldsymbol{\theta}_t^{(i)}) = \frac{\sqrt[3]{n}\, \exp(-2|(y_t^i - y_t)\, \sqrt[3]{n}|)}{\sum_{i=1}^{n} \sqrt[3]{n}\, \exp(-2|(y_t^i - y_t)\, \sqrt[3]{n}|)}, \tag{3.12}$$

donde, y_t^i es la partícula de la variable observada, y_t es el valor observado y n es el número de partículas.

Algoritmo 3.1 Filtro de partículas Liu-West (LW)

1: **Para** $t = 1, \ldots, T$ **hacer**

2: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_{t-1}^{(i)}, \boldsymbol{\theta}_{t-1}^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{w_t^{(i)} = p(y_t \,|\, g(x_{t-1}^{(i)}), m_{t-1}^{(i)}) : i = 1, \ldots, N\}$.

3: Obtener muestras $\boldsymbol{\theta}_t^{(i)}$ del KS,

$$\boldsymbol{\theta}_t^{(i)} \sim N(\boldsymbol{\theta} \,|\, \tilde{m}_{t-1}^{(i)}, h^2 V_{t-1})$$

4: Obtener muestras $x_t^{(i)}$ de la distribución por importancia,

$$x_t^{(i)} \sim p(x_t \,|\, \tilde{x}_{t-1}^{(i)}, \boldsymbol{\theta}_t^{(i)}), \quad i = 1, \ldots, N.$$

5: Calcular los pesos nuevos

$$\omega_t^{(i)} \propto \frac{p(y_t \,|\, x_t^{(i)}, \boldsymbol{\theta}_t^{(i)})}{p(y_t \,|\, g(\tilde{x}_{t-1}^{(i)}), \tilde{m}_{t-1}^{(i)})},$$

y normalizarlos para que sumen 1.

6: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_t^{(i)}, \boldsymbol{\theta}_t^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{\omega_t^{(i)} : i = 1, \ldots, N\}$, luego tomar $\omega_t^{(i)} = 1/N$.

7: **Fin Para**

3.4.3. Filtro de partículas marginal Metropolis-Hastings

Uno de los algoritmos que se consideró fue el filtro de partículas marginal Metropolis-Hastings [1]. Este algoritmo aproxima la distribución posterior $p(\boldsymbol{\theta} \,|\, y_{0:T})$ usando la combinación de los métodos: el filtro de partículas [12] y las cadenas de Markov vía Monte Carlo [17, 30]. El primero, es empleado para aproximar $p(x_T \,|\, y_{0:T}, \boldsymbol{\theta})$ y la verosimilitud $p(y_{0:T} \,|\, \boldsymbol{\theta})$. El segundo, se usa para aproximar $p(\boldsymbol{\theta} \,|\, y_{0:T})$ basado en la verosimilitud obtenida por el filtro de partículas [11].

Algoritmo 3.2 Filtro de partículas marginal Metropolis-Hastings (PMMH)

1: Obtener muestra de la distribución apriori:

$$\boldsymbol{\theta}_0 \sim p(\boldsymbol{\theta}).$$

2: Ejecutar el filtro de partículas, Algoritmo 2.2, con N partículas para obtener $\tilde{p}(y_{0:T} \,|\, \boldsymbol{\theta}_0)$.
3: **Para** $m = 1, \ldots, M$ **hacer**
4: Propuesta: $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_{m-1})$.
5: Ejecutar el filtro de partículas, Algoritmo 2.2, con N partículas para obtener $\tilde{p}(y_{0:T} \,|\, \boldsymbol{\theta}^*)$.
6: Calcular:

$$\alpha = \text{mín}\left(1, \frac{p(\boldsymbol{\theta}^*)\, \tilde{p}(y_{0:T} \,|\, \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}_{m-1})\, \tilde{p}(y_{0:T} \,|\, \boldsymbol{\theta}_{m-1})}\right).$$

7: Hacer

$$\boldsymbol{\theta}_m = \begin{cases} \boldsymbol{\theta^*} & \text{con probabilidad } \text{mín}(\alpha, 1) \\ \boldsymbol{\theta}_{t-1} & \text{con probabilidad } 1 - \text{mín}(\alpha, 1). \end{cases}$$

8: **Fin Para**

La expresión para la tasa α (tasa de aceptación), en el Algoritmo 3.2 punto 6, sugiere que el algoritmo efectivamente aproxima la densidad marginal $p(\boldsymbol{\theta} \,|\, y_{0:T})$, justificando el termino marginal [1].

El atractivo de este algoritmo es que la dificultad de muestrear de $p(x_{0:T},\ \boldsymbol{\theta} \,|\, y_{0:T})$ es reducida a muestrear en $p(\boldsymbol{\theta} \,|\, y_{0:T})$ [1]. Queda claro que este algoritmo se enfoca en estimar los parámetros del modelo en vez de estimar la variable latente.

Para aplicar el filtro bootstrap, Algoritmo 2.2, se usó las distribuciones (3.9) y (3.10) para los valores iniciales. Para obtener los pesos w se utilizó la medida de probabilidad normal con promedio y_t y desviación estándar 1, esto es

$$w_t^{(i)} \sim N(y_t, 1).$$

Se usó esta probabilidad porque se obtuvieron mejores resultados que al usar la pro-

babilidad dada por (3.12).

Para la distribución a priori y la propuesta se utilizaron distribuciones normales multivariadas truncadas, con valor mínimo $(0, 0, 0, 0)$ y valor máximo $(1, 1, 1, 1)$. La distribución a priori usa la esperanza $(0, 0, 0.75, 0)$ y la matriz de covarianza es una matriz diagonal con elementos $(0.001, 0.01, 0.01, 0.0000015)$. Para la distribución propuesta se usa una matriz de covarianza, una matriz diagonal con valores $(0.00001, 0.001, 0.001, 0.0000001)$. Se decidió usar estos valores pues al usar otros valores los resultados del ajuste no varían mucho.

3.4.4. Filtro de partículas secuencial aumentado MCMC

Se implementó el algoritmo propuesto en [20]. El algoritmo es una combinación de los métodos:

- Secuenciales, como LW, en donde el muestreo permite actualizaciones sucesivas de la distribución posterior a medida que hay nuevas observaciones disponibles,
- y los métodos que usan MCMC, que permiten obtener muestras de la distribución objetivo posterior de $\boldsymbol{\theta}$, además que permiten obtener propuestas arbitrarias sin tener que asumir la existencia de estadísticos suficientes o un modelo artificial para la evolución de $\boldsymbol{\theta}$.

El algoritmo propuesto en [20] construye muestras por medio de MCMC para cada $t = 1, \ldots, T$, en las cuales sólo incorpora el t-ésimo componente $p(y_t \mid y_{1:t-1}, \boldsymbol{\theta})$ de la verosimilitud. Además, asume que la distribución posterior $p(\boldsymbol{\theta} \mid y_{0:t})$ puede ser aproximada por una mixtura gaussiana. La mixtura gaussiana es obtenida a partir de la muestra $({}^{\mathbf{1}}\boldsymbol{\theta}_t^B, \ldots, {}^{\boldsymbol{M}}\boldsymbol{\theta}_t^B)$ recolectada después del periodo de quema. El artículo [20] realiza el ajuste para $(\boldsymbol{\theta}_t^B, x_t)$.

Para determinar el número de componentes de la mixtura se realizaron pruebas con diferentes valores. Se obtuvieron resultados buenos con un componente, es decir con

una normal multivariada truncada con parámetros promedio y la matriz de covarianza de la muestra.

Algoritmo 3.3 Filtro de partículas secuencial aumentado MCMC

1: Iniciar con M valores iniciales $(\boldsymbol{\theta}_0^m,\, x_0^m)$ de la densidad a priori $p(\boldsymbol{\theta}, x_0)$.
2: **Para** $t = 1, \ldots, T$ **hacer**
3: Ajustar modelo $p(\boldsymbol{\theta} \,|\, y_{0:t})$ con $\{{}^{\mathbf{1}}\boldsymbol{\theta}_{t-1}^B, \ldots, {}^{\boldsymbol{M}}\boldsymbol{\theta}_{t-1}^B\}$
4: **Para** $m = 1, \ldots, M$ **hacer**
5: **Para** $k = 1, \ldots, B$ **hacer**
6: Generar

$$\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \,|\, \boldsymbol{\theta}^{k-1})$$

7: Obtener

$$x_t^* \sim p(x_t^* \,|\, x_{t-1},\, \boldsymbol{\theta}^*)$$

8: Calcular

$$\alpha = \text{mín}\left(1, \frac{p(y_t \,|\, x_t^*, \boldsymbol{\theta}^*)\, \hat{p}(\boldsymbol{\theta}^* \,|\, y_{0:t-1})}{p(y_t \,|\, x_t^k, \boldsymbol{\theta}^k)\, \hat{p}(\boldsymbol{\theta}^k \,|\, y_{0:t-1})}\right)$$

9: Hacer

$$\boldsymbol{\theta}^k = \begin{cases} \boldsymbol{\theta^*} & \text{con probabilidad } \text{mín}(\alpha, 1) \\ \boldsymbol{\theta}^{k-1} & \text{con probabilidad } 1 - \text{mín}(\alpha, 1). \end{cases}$$

10: **Fin Para**
11: **Fin Para**
12: Recolectar $\{{}^{\mathbf{1}}\boldsymbol{\theta}_t^B, \ldots, {}^{\boldsymbol{M}}\boldsymbol{\theta}_t^B\}$
13: Generar $x_t^m \sim p(x_t \,|\, x_{t-1}^m,\, \boldsymbol{\theta}_t^m)$
14: **Fin Para**

Para los valores iniciales de x_0 y y_0 se usaron las distribuciones (3.9) y (3.10). Para la obtención de los pesos $w_t = p(y_t \,|\, x_t, \boldsymbol{\theta})$ se utilizó la medida de probabilidad normal con promedio y_t y desviación estándar 1, esto es $N(\cdot \,|\, y_t, 1)$. Se usó esta probabilidad porque se obtuvieron mejores resultados que al usar la probabilidad dada por (3.12).

Para obtener los valores iniciales de $\boldsymbol{\theta}_0$ se usó la distribución normal multivariada truncada con valor mínimo $(0, 0, 0, 0)$ y valor máximo $(1, 1, 1, 1)$, con esperanza $(0, 0, 0.75, 0)$, como matriz de covarianza una matriz diagonal con elementos

$(0.0005, 0.001, 0.05, 0.0000015)$. La distribución propuesta q que se utilizó fue una distribución normal multivariada truncada con valor mínimo $(0, 0, 0, 0)$ y valor máximo $(1, 1, 1, 1)$, cuya matriz de covarianza es una matriz diagonal con elementos $(0.06, 0.05, 0.05, 0.05)$. Se decidió usar estos valores pues al usar otros valores los resultados del ajuste no varían mucho.

Este algoritmo permite realizar su implementación en paralelo. El ciclo que se realiza sobre el número de partículas, cada iteración corresponde a la construcción de la cadena de Markov de la correspondiente partícula m y es independiente de las otras. Se usó la función *mclapply* del paquete parallel[1] de R. El tiempo de ejecución se redujo en un 80 % con respecto al tiempo de una implementación estándar de este mismo algoritmo.

3.4.5. Filtro de partículas secuencial aumentado MCMC con remuestreo

En esta versión del Algoritmo 3.3 filtro de partículas secuencial aumentado MCMC se le agregó el paso de remuestreo de las muestras, con el propósito de mejorar la estimación de los caminos de la variable latente y la variable observada. Las líneas del Algoritmo 3.4 se agrega al Algoritmo 3.3 después de la línea 13.

Algoritmo 3.4 Remuestreo en SAMCMC

1: Obtener los pesos

$$\omega_t^{(i)} \sim p(y_t \mid x_t^{(i)}, \boldsymbol{\theta}_t^{(i)})$$

2: Realizar remuestreo con reemplazo de tamaño N del conjunto $\{x_t^{(i)}, \boldsymbol{\theta}_t^{(i)} : i = 1, \ldots, N\}$ con probabilidades $\{\omega_t^{(i)} : i = 1, \ldots, N\}$.

Para la obtención de los pesos se usa la medida de probabilidad (3.12). El remuestreo se realiza con el método usado en [19] llamado remuestreo sistemático, el cual tiene mejor varianza que el simple remuestreo aleatorio, además se espera que tenga un mejor

[1] Versión 3.6.3 del paquete parallel.

desempeño que el remuestreo estratificado [16]. Es un remuestreo popular por su fácil implementación y por su baja complejidad computacional. El Algoritmo 3.5 muestra el procedimiento.

Algoritmo 3.5 Remuestreo sistemático

1: Obtener $u \sim U([0,1])$
2: Normalizar los pesos $\omega^{(i)}$
3: Hacer $\overline{u} = u/N$, $j = 1$ y $S_w = \omega^1$.
4: **Para** $K = 1, \ldots, n$ **hacer**
5: **Mientras** $S_w < \overline{u}$ **hacer**
6: Hacer $j = j + 1$ y $S_w = S_w + \omega^j$.
7: **Fin Mientras**
8: Hacer $a^k = j$ y $\overline{u} = \overline{u} + 1/N$.
9: **Fin Para**
10: **Devolver** $a^{1:N}$

Al remuestreo se le agrega una restricción debido a que el número de elementos únicos en la muestra de los parámetros se va reduciendo. La restricción consiste en aplicar el remuestreo si el número de elementos únicos en la muestra de los parámetros al realizar el remuestreo es mayor al 50 % del número total de partículas.

CAPÍTULO 4

RESULTADOS

Una vez que se contó con la información simulada y real, se procedió a aplicar los algoritmos y mostrar la eficacia en la estimación de los parámetros del modelo.

El capítulo se desarrolla de la siguiente manera: en la sección 4.1 se muestran los resultados del algoritmo filtro de partículas Liu-West (LW), en la sección 4.2 los resultados del algoritmo filtro de partículas Marginal Metropolis-Hastings (PMMH), la sección 4.3 contiene resultados del filtro Secuencial Aumentado MCMC (SAMCMC) además de pruebas de sensibilidad en el periodo de quema y número de partículas, sección 4.4 se presentan los resultados del algoritmo SAMCMC agregándole remuestreo, en la sección 4.5 se comparan los resultados y se determina cual algoritmo es mejor y finalmente la sección 4.6 muestra los resultados de ejecutar SAMCMC en los datos reales.

4.1. Filtro de partículas Liu-West

El algoritmo de Liu-West (LW) es el punto de referencia para los otros dos algoritmos que se usaron en el estudio. El algoritmo usa 130 partículas, sus resultados

se presentan a continuación. En el Cuadro 4.1 se puede ver el resumen de valores estadísticos de la muestra obtenida.

Cuadro 4.1: *Resumen de valores estadísticos de las muestras obtenidas con LW.*

	α	β	H	μ
Min.	0.1883	0.1911	0.4296	0.1760
5 % Per.	0.1889	0.1926	0.4303	0.1775
1st Qu.	0.1902	0.1965	0.4316	0.1815
Median	0.1909	0.2049	0.4323	0.1899
Mean	0.1922	0.2022	0.4337	0.1872
3rd Qu.	0.1934	0.2060	0.4351	0.1910
95 % Per.	0.1984	0.2080	0.4401	0.1930
Max.	0.1997	0.2085	0.4414	0.1935

El algoritmo generó muestras con poca variabilidad, pero la estimación de los parámetros del modelo no es aceptable, ni siquiera los valores reales se encuentran en el rango de las muestras, tal como esto se puede apreciar en los Cuadros 4.1 y 4.2.

Cuadro 4.2: *Valor promedio, desviación estándar y coeficiente de variación de la estimación de los parámetros junto con el valor real a estimar con LW.*

	α	β	H	μ
Mean	0.19224	0.20215	0.43374	0.18715
Sd	0.00301	0.00531	0.00314	0.00533
CV	0.01566	0.02627	0.00724	0.02848
Real	0.02733	0.07567	0.60000	0.00140

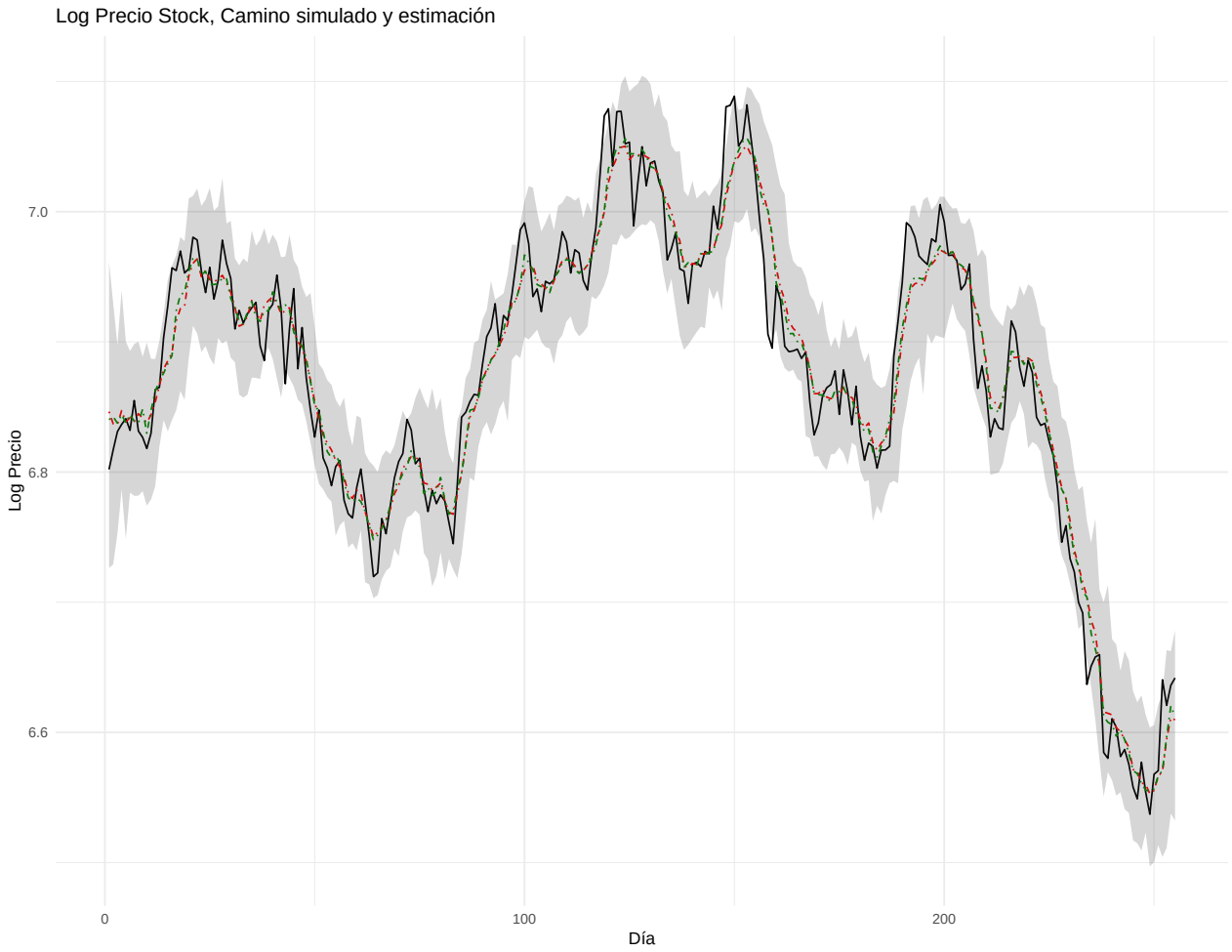


Figura 4.1: *Camino del Log Precio Stock simulado. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde es mediana, banda gris intervalo de confianza 90 %. Algoritmo LW*

A diferencia de la estimación de los parámetros, la estimación de la variable observada Log Precio es muy buena. El algoritmo LW logró reconstruir muy bien el camino observado como se aprecia en la Figura 4.1 y se aprecia además que la banda de confianza a un 90 %, área sombreada color gris, es angosta. Se visualizan un par de observaciones que no entran en la banda pero aún así la estimación es aceptable.

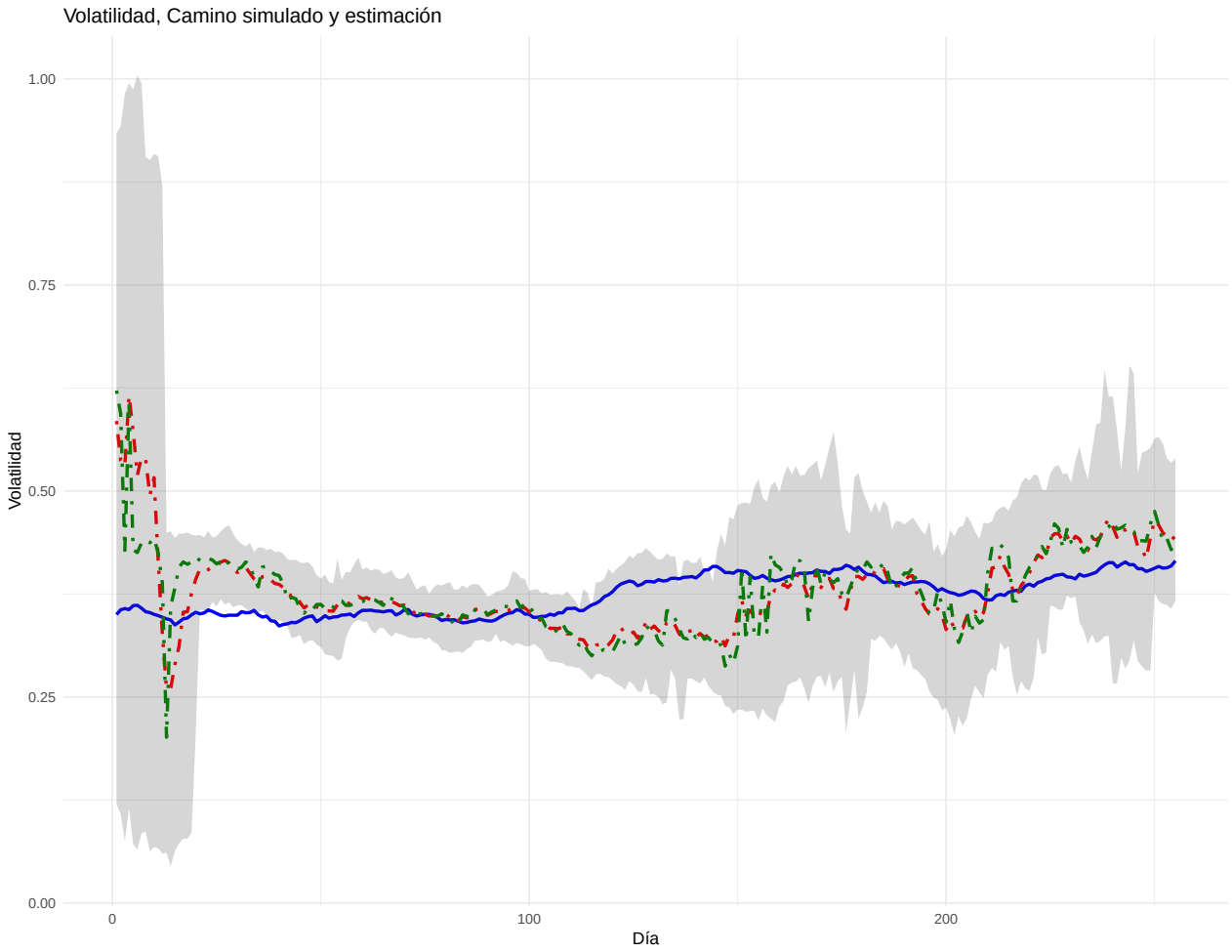


Figura 4.2: *Camino de la volatilidad simulada línea color azul. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde es mediana, banda gris intervalo de confianza 90 %. Líneas grises muestra de caminos estimados. Algoritmo LW*

En cuanto a la estimación del camino de la volatilidad simulada, la cual se muestra en la Figura 4.2, vemos que el algoritmo no reconstruye muy bien la volatilidad observada en los primeros 60 días aproximadamente pero después la estimación es buena. Tiene una banda de confianza que al inicio es muy ancha pero luego disminuye y mantiene un ancho relativamente fijo. Por otro lado, el camino real se sale de la banda en pequeños tramos.

Las medidas de la estimación, el RECM, $\text{IS}_{90\,\%}$ y $\text{IS}_{80\,\%}$ se presentan en el Cuadro

4.3, más adelante se usarán para comparar con los otros algoritmos.

Cuadro 4.3: *Raíz del error cuadrático medio (RECM) e Interval score (IS) con niveles 80 % y 90 % de las variables del modelo. Algoritmo LW.*

Variable	RECM	$IS_{90\,\%}$	$IS_{80\,\%}$
Volatilidad	0.16505	0.14641	0.11160
Log Precio	0.04263	1.42452	0.86378
α	0.10797	2.09150	1.05615
β	0.06824	1.27547	0.64866
H	0.27886	5.50899	2.76288
μ	0.13470	2.60580	1.31392

El tiempo promedio de las 10 ejecuciones es de 6.60 minutos.

4.1.1. Sensibilidad al número de partículas

El algoritmo no genera estimaciones aceptables de los parámetros del modelo, pero se realizaron escenarios variando el número de partículas para ver que efectos tienen en los resultados. Los resultados se pueden observar en los Cuadros 4.4, 4.5.

Cuadro 4.4: *Coeficiente de variación de los escenarios de sensibilidad al número de partículas en LW.*

Partículas	α	β	H	μ
130	1.57 %	2.63 %	0.72 %	2.85 %
500	8.92 %	5.52 %	15.82 %	6.97 %
1,000	5.94 %	3.21 %	18.86 %	4.70 %
10,000	9.39 %	13.13 %	30.39 %	8.22 %

Podemos ver que:

- En general es un algoritmo que consume poco tiempo en su ejecución.
- El aumento en el número de partículas no mejora las estimaciones.
- Las muestras se vuelven más heterogéneas conforme aumenta el número de partículas, es decir hay más variabilidad en las muestras.

Cuadro 4.5: *Valores promedio y desviación estándar para diferentes escenarios de número de partículas en LW.*

Partículas	Tiempo (Minutos)		α	β	H	μ
130	6.75	mean	0.19224	0.20215	0.43374	0.18715
		sd	0.00301	0.00531	0.00314	0.00533
500	25.69	mean	0.14205	0.14414	0.35632	0.13709
		sd	0.01267	0.00795	0.05636	0.00955
1,000	52.32	mean	0.16584	0.17957	0.40489	0.16506
		sd	0.00985	0.00577	0.07635	0.00776
10,000	528.83	mean	0.15171	0.14915	0.43330	0.14781
		sd	0.01425	0.01958	0.13168	0.01215
	Real		**0.02733**	**0.07567**	**0.60000**	**0.00140**

4.2. Filtro de partículas Marginal Metropolis-Hastings

El algoritmo filtro de partículas Marginal Metropolis-Hastings (PMMH) se ejecutó con un tamaño de muestra de 10,000 y se quemó[1] el 50 %. Se usaron distribuciones normales para las distribuciones a priori y la propuesta.

Como se puede apreciar en el Cuadro 4.6 y Figura 4.3, las muestras tienen histogramas asimétricos. Se ve la gran influencia de la distribución a priori, la cual tiene

[1]Periodo en el cual la cadena de Markov se mueve de su posición inicial a la región del espacio de parámetros que tiene alta probabilidad posterior.

una media $(\alpha, \beta, H, \mu) = (0, 0, 0.75, 0)$. Se ve en la Figura 4.3 que los valores reales de (α, β, μ) se encuentran en zonas de alta probabilidad en las distribuciones empíricas de las muestras. Para el parámetro H su valor real se encuentra en una zona con probabilidad no tan alta en la distribución empírica de la muestra.

Cuadro 4.6: *Resumen de valores estadísticos de las muestras obtenidas con PMMH.*

	α	β	H	μ
Min.	0.000001	0.0000654	0.3845	0.0000007
5 % Per.	0.00446	0.0111764	0.5859	0.000128
1st Qu.	0.01465	0.0439644	0.6869	0.000520
Median	0.02746	0.0765881	0.7496	0.000979
Mean	0.03116	0.0884717	0.7461	0.001143
3rd Qu.	0.04513	0.1177788	0.8102	0.001525
95 % Per.	0.06824	0.2074704	0.8952	0.002806
Max.	0.08825	0.3571898	0.9951	0.004197

El valor estimado de los parámetros, desviación estándar y coeficiente de variación se muestra en el Cuadro 4.7. Se obtienen estimaciones buenas, a excepción del parámetro H. Con respecto a la variabilidad, en general es alta.

Cuadro 4.7: *Valor promedio, desviación estándar y coeficiente de variación de la estimación de los parámetros junto con el valor real a estimar con PMMH.*

	α	β	H	μ
Mean	0.03116	0.08847	0.74609	0.00114
Sd	0.01998	0.06209	0.09437	0.00082
CV	0.64121	0.70182	0.12649	0.71930
Real	0.02733	0.07567	0.60000	0.00140

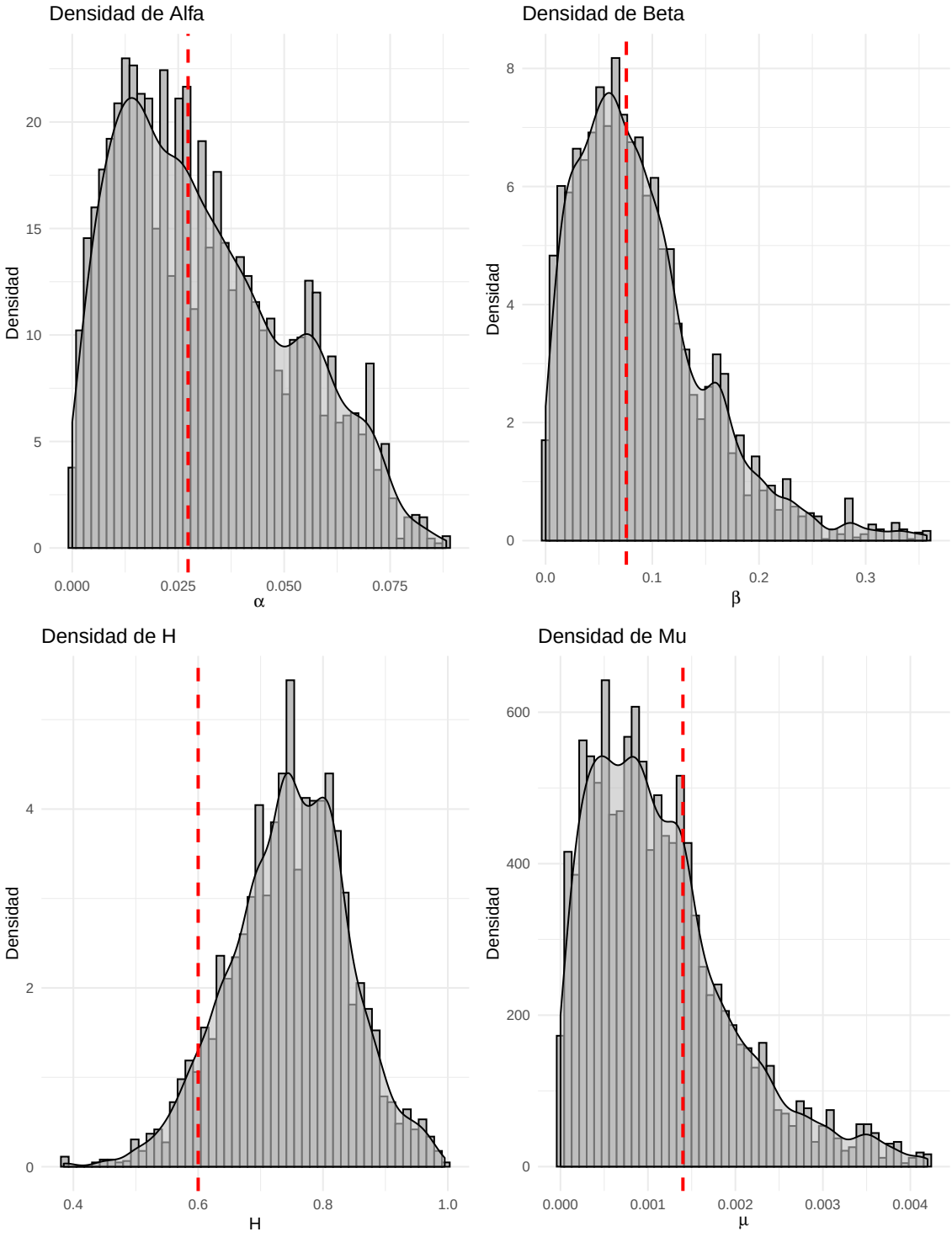


Figura 4.3: *Histogramas de las muestras obtenidas con PMMH. La línea discontinua de color rojo muestra el valor real. El área sombreada color gris corresponde a la estimación de la densidad por medio de un kernel gaussiano.*

La Figura 4.3 une la información de los dos cuadros anteriores. Exhibe los histogramas y la área sombreada que permite visualizar la densidad empírica de las distribuciones. Una línea roja vertical muestra donde se ubica el valor real dentro de la densidad. No todos los valores reales se ubican en zonas de alta probabilidad posterior, algo esperado en una estimación aceptable.

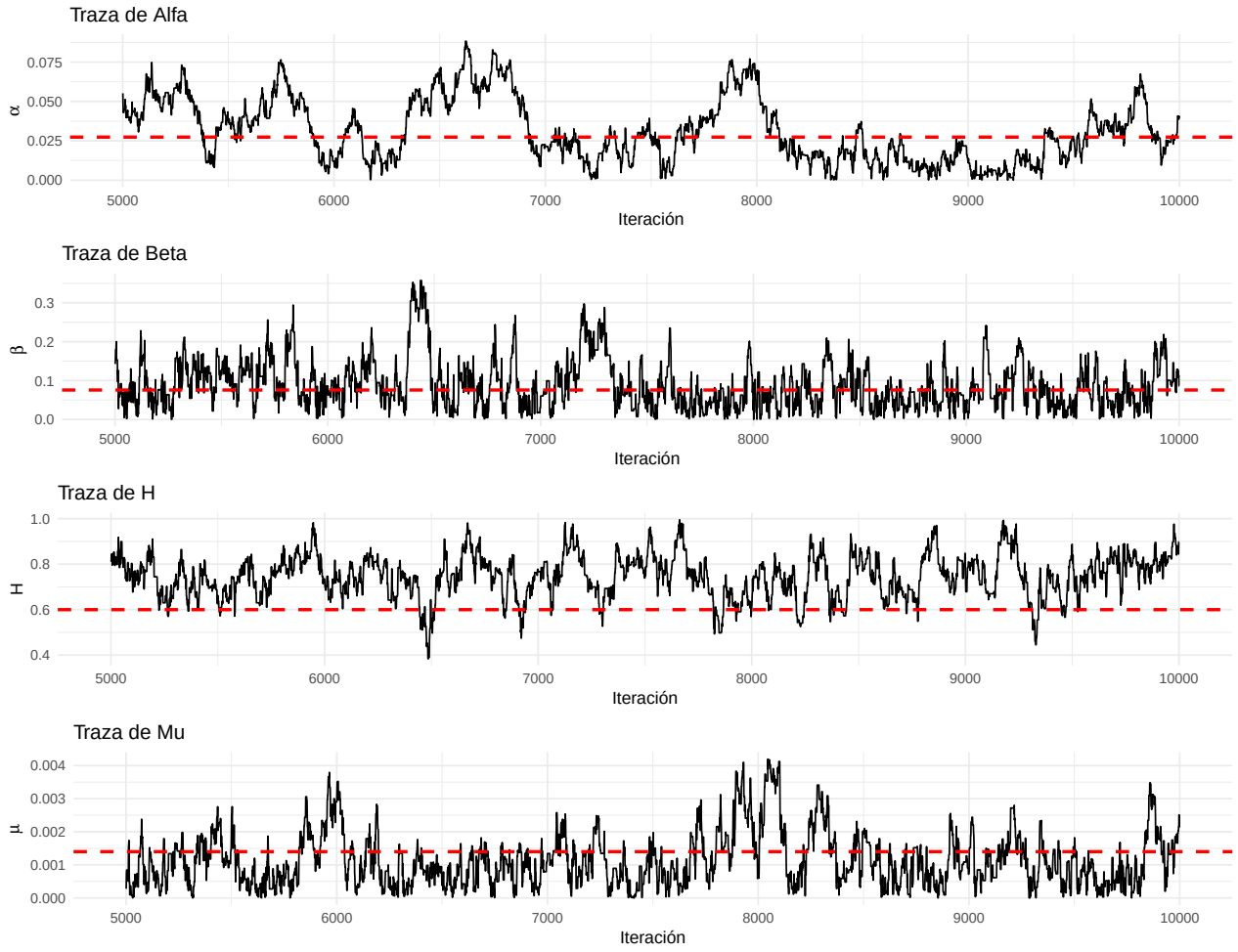


Figura 4.4: *Gráfico de traza de las muestras obtenidas con PMMH. La línea discontinua de color rojo muestra el valor real.*

Los gráficos de traza del MCMC[2] para cada parámetro se ve en la Figura 4.4. La traza de α da evidencia de tener ciertas tendencias y aunque se quemaron las

[2]Siglas del inglés Markov Chain Monte Carlo.

primeras 5,000 iteraciones, no parece tener una buena mezcla. Los procesos de los otros parámetros tienen en general buena mezcla.

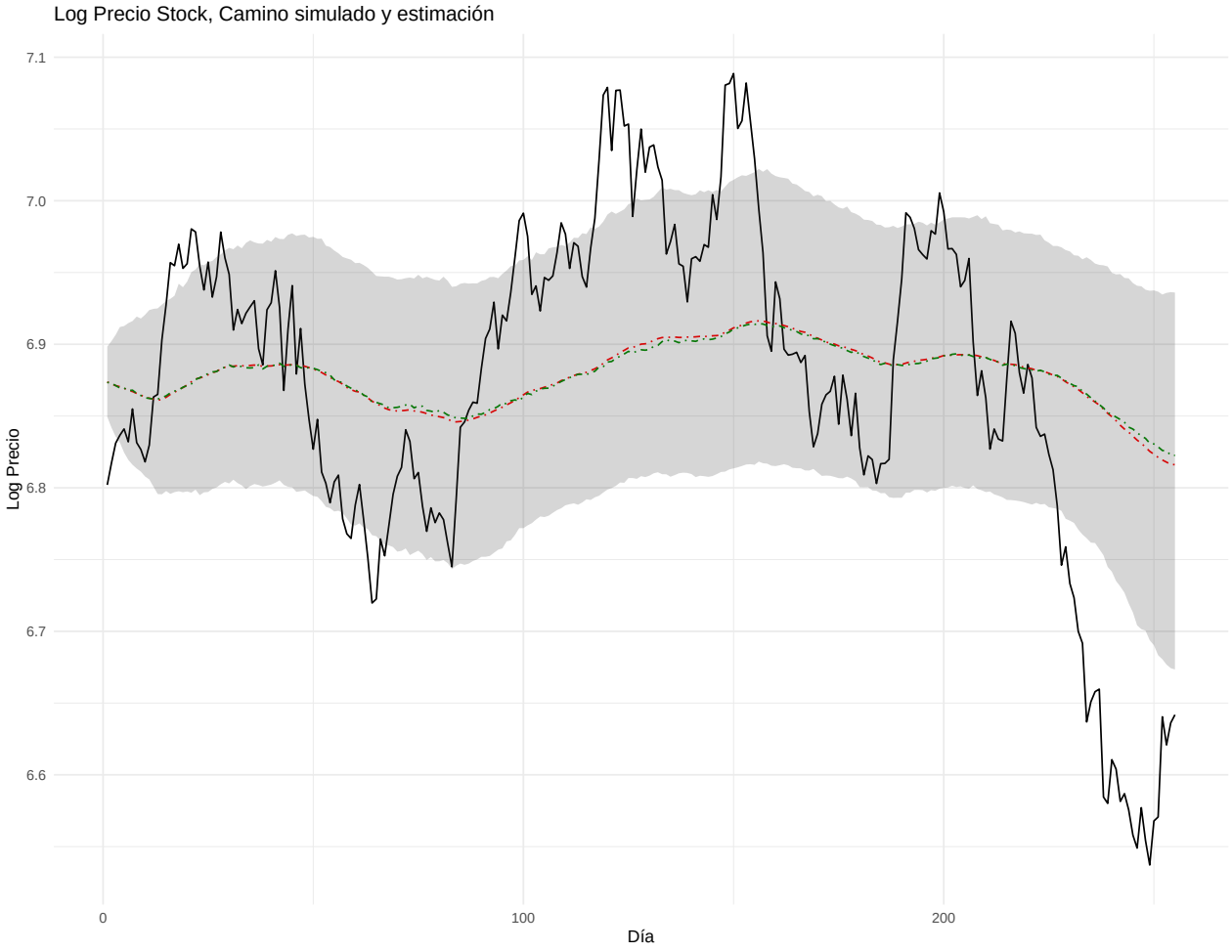


Figura 4.5: *Camino del Log Precio Stock simulado. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde es mediana, banda gris intervalo de confianza 90 %. Algoritmo PMMH.*

En la Figura 4.5 se aprecia que la reconstrucción del camino del Log Precio no es buena, pues hay varias secciones del camino que quedan por fuera de la banda de 90 % de confianza. De forma adicional, se observa que prácticamente no hay diferencia entre usar el promedio o la mediana como estimador.

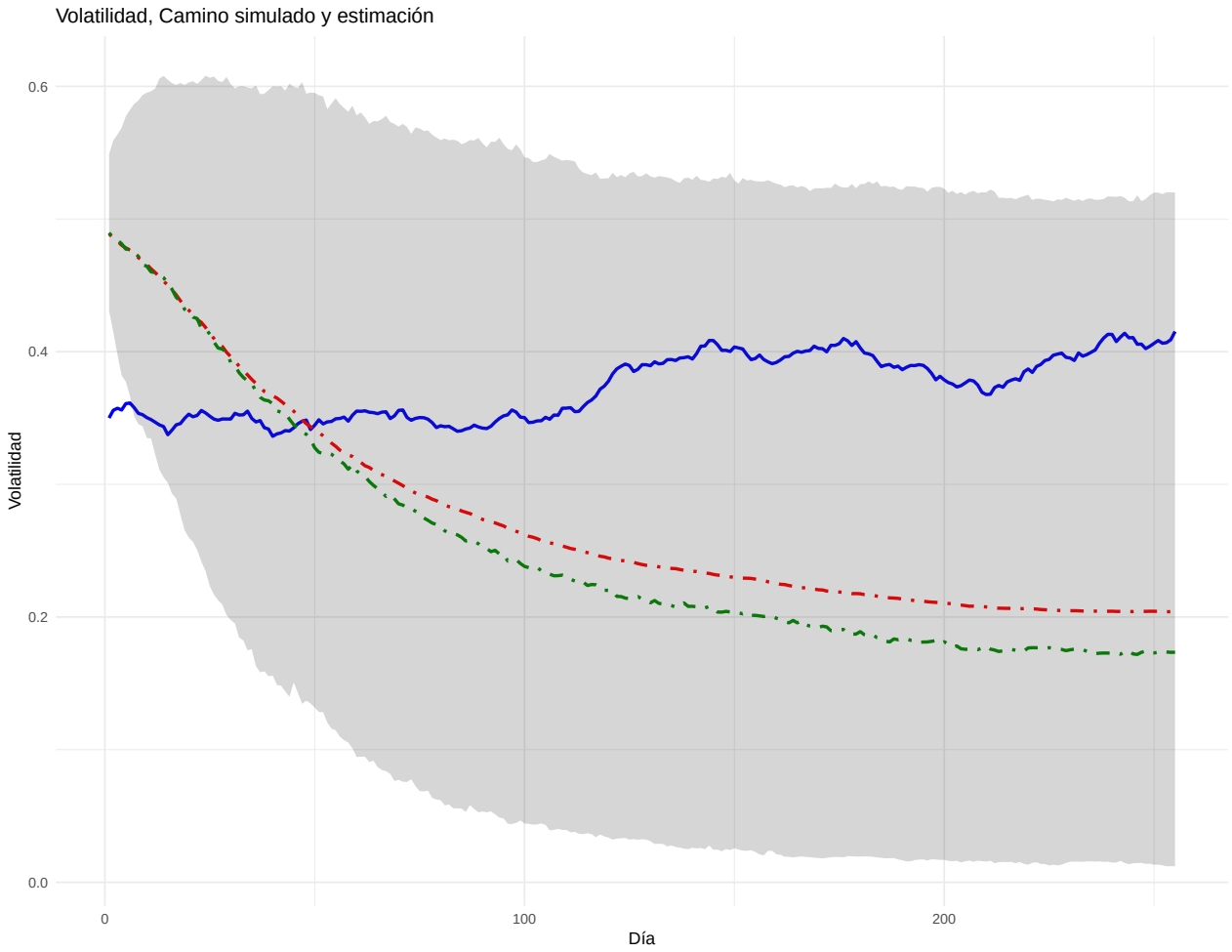


Figura 4.6: *Camino de la volatilidad simulada línea color azul. Línea discontinua roja valor promedio de la estimación, banda gris intervalo de confianza 90 %. Algoritmo PMMH.*

Al igual que en los otros algoritmos, la estimación del camino de la volatilidad no es buena. Como se aprecia en la Figura 4.6 la estimación decrece con el tiempo, situación contraria al camino real. La banda de confianza es ancha prácticamente desde el inicio hasta el final del camino.

Cuadro 4.8: *Raíz del error cuadrático medio (RECM) e Interval score (IS) con niveles 80 % y 90 % de las variables del modelo. Algoritmo PMMH.*

Variable	RECM	$IS_{90\,\%}$	$IS_{80\,\%}$
Volatilidad	0.20129	0.55102	0.41654
Log Precio	0.10597	0.47259	0.37931
α	0.01777	0.05471	0.04313
β	0.06295	0.19630	0.15490
H	0.17194	0.38390	0.52944
μ	0.00082	0.00234	0.00183

Las medidas del desempeño del algoritmo se muestran en el Cuadro 4.8. Se visualizan valores altos para la volatilidad, log Precio y H. En la sección 4.5 se verá el desempeño en relación a los otros algoritmos.

El tiempo promedio de las 10 ejecuciones es de 11.84 horas.

4.2.1. Distribución a priori Gamma

Se realizó un escenario en el cual se cambió la distribución conjunta a priori de los parámetros, con el fin de ver si la influencia en la estimación se sigue presentando. Se escogió una distribución gamma con con parámetro de forma $(\alpha, \beta, H, \mu) = (5, 13, 100, 2)$ y parámetro de tasa $(\alpha, \beta, H, \mu) = (150, 150, 150, 1500)$. Con estos parámetros la esperanza y la desviación estándar son $(\alpha, \beta, H, \mu) = (0.0333, 0.0867, 0.6667, 0.0013)$ y $(\alpha, \beta, H, \mu) = (0.0149, 0.0240, 0.0667, 0.0009)$ respectivamente.

Como se puede ver en los Cuadros 4.9 y 4.10, el promedio y la desviación estándar de las muestras de los parámetros prácticamente son los de la distribución a priori. Esto se puede ver como una debilidad del algoritmo, pues si no se cuenta con información a priori se tiene que hacer uso de distribuciones no informativas [30] que no van a ayudar a obtener buenas estimaciones.

Cuadro 4.9: *Resumen de valores estadísticos de las muestras obtenidas con PMMH Gamma.*

	α	β	H	μ
Min.	0.005424	0.03028	0.4700	0.000011
5 % Per.	0.016273	0.05348	0.5551	0.000239
1st Qu.	0.023986	0.07031	0.6184	0.000688
Median	0.034849	0.08375	0.6589	0.001268
Mean	0.036093	0.08666	0.6571	0.001566
3rd Qu.	0.046388	0.10153	0.6974	0.002351
95 % Per.	0.061761	0.12619	0.7559	0.003699
Max.	0.073953	0.16053	0.8859	0.005089

Cuadro 4.10: *Valor promedio, desviación estándar y coeficiente de variación de la estimación de los parámetros junto con el valor real a estimar con PMMH Gamma.*

	α	β	H	μ
Mean	0.03609	0.08666	0.65711	0.00156
Sd	0.01454	0.02261	0.06007	0.00109
CV	0.40288	0.26090	0.09142	0.69872
Real	0.02733	0.07567	0.60000	0.00140

4.3. Filtro de partículas Secuencial Aumentado MCMC

En esta sección se presenta los resultados de aplicar el algoritmo filtro de partículas Secuencial Aumentado MCMC (SAMCMC) en los datos simulados.

El algoritmo usa 130 partículas que es equivalente al número de MCMC's que se ejecutan en cada iteración de t. Para cada MCMC que realiza un periodo de quema de 1,000 iteraciones.

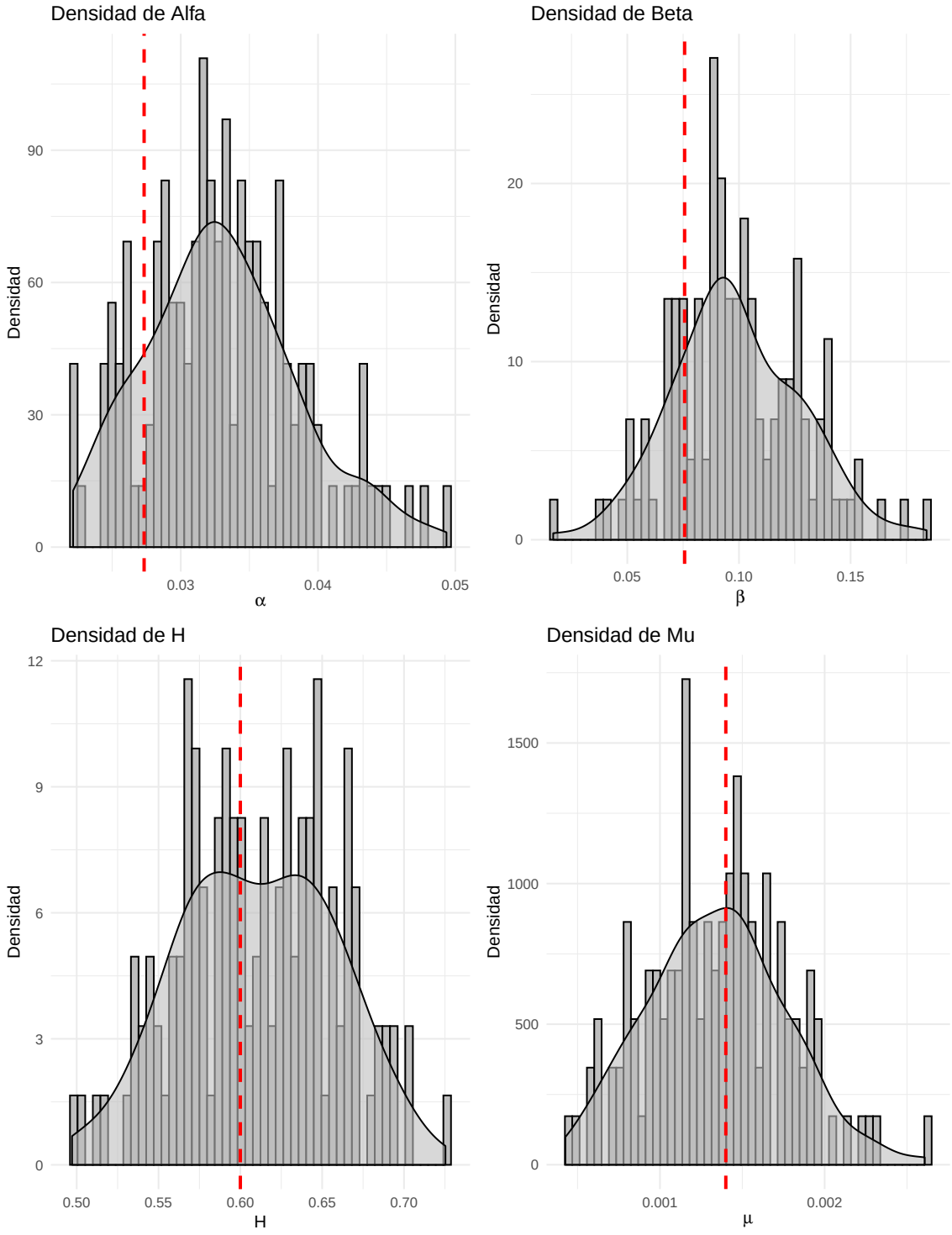


Figura 4.7: *Histogramas de las muestras obtenidas con SAMCMC. La línea discontinua de color rojo muestra el valor real. El área sombreada color gris corresponde a la estimación de la densidad por medio de un kernel gaussiano.*

El principal resultado del algoritmo es una muestra de la distribución posterior de cada uno de los parámetros del modelo de espacio estado (2.11), (2.12). En el Cuadro 4.11 se muestra un resumen de valores estadísticos de la muestra de cada uno de los 4 parámetros. El cuadro da evidencia de que las distribuciones son simétricas.

Cuadro 4.11: *Resumen de valores estadísticos de las muestras obtenidas con SAMCMC.*

	α	β	H	μ
Min.:	0.02216	0.01682	0.4972	0.0004258
5 % Pe.:	0.02470	0.05347	0.5349	0.0006655
1st Qu.:	0.02898	0.08060	0.5729	0.0010783
Median:	0.03259	0.09515	0.6109	0.0013529
Mean:	0.03299	0.09884	0.6103	0.0013438
3rd Qu.:	0.03593	0.11969	0.6457	0.0016134
95 % Pe.:	0.04326	0.14456	0.6868	0.0019719
Max.:	0.04936	0.18402	0.7253	0.0026080

Cuadro 4.12: *Valor promedio y desviación estándar de la estimación de los parámetros junto con el valor real a estimar con SAMCMC.*

	α	β	H	μ
Mean	0.03299	0.09884	0.61030	0.00134
Sd	0.00562	0.02888	0.04750	0.00042
CV	0.17035	0.29219	0.07783	0.31343
Real	0.02733	0.07567	0.6000	0.00140

Como los datos son simulados, se conocen los valores reales de los parámetros que los generaron. El Cuadro 4.12 presenta el valor estimado (el valor promedio de la muestra), la desviación estándar y el coeficiente de variación para los parámetros. Los valores

estimados son aceptables, el parámetro β es el que está un poco lejos del valor real y el parámetro μ es el que tiene mayor variabilidad con respecto a los otros parámetros.

La Figura 4.7 se aprecia que los valores reales se ubican en zonas de alta probabilidad posterior, algo esperado en una estimación aceptable.

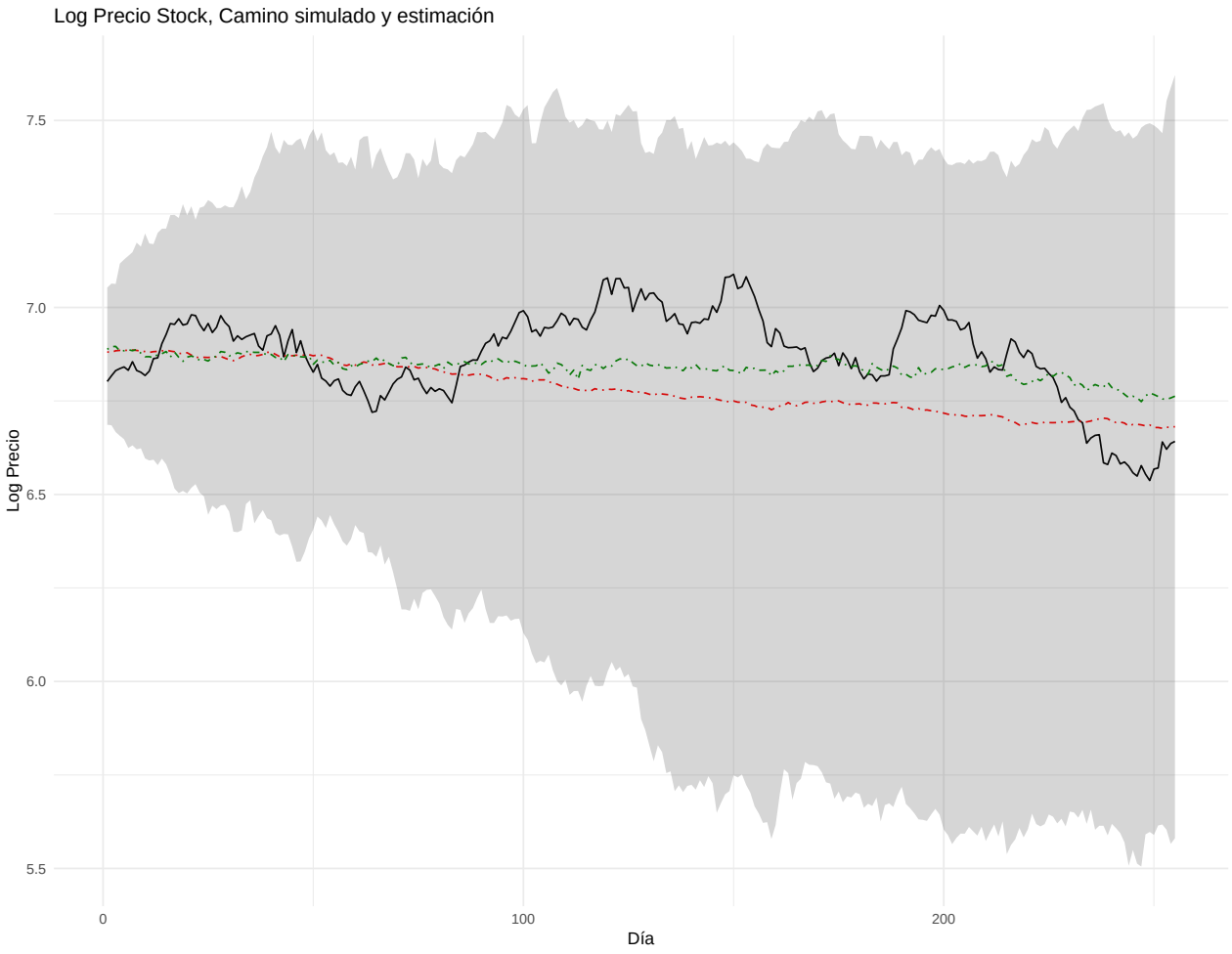


Figura 4.8: *Camino del Log Precio Stock simulado. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde es mediana, banda gris intervalo de confianza 90 %. Algoritmo SAMCMC.*

El objetivo principal es la estimación de los parámetros del modelo, pero también se puede ver como el algoritmo se desempeñó reconstruyendo el camino del logaritmo de los precios. En la Figura 4.8, la línea negra es el camino simulado del logaritmo de los

precios. La línea rojo discontinua es el camino de los promedios para cada día, esto pues para cada día hay una muestra de la distribución posterior de la variable observada. La banda gris representa un intervalo de confianza de 90 %, que se obtiene con los intervalos diarios. La banda de confianza tiende a ensancharse conforme aumenta el tiempo.

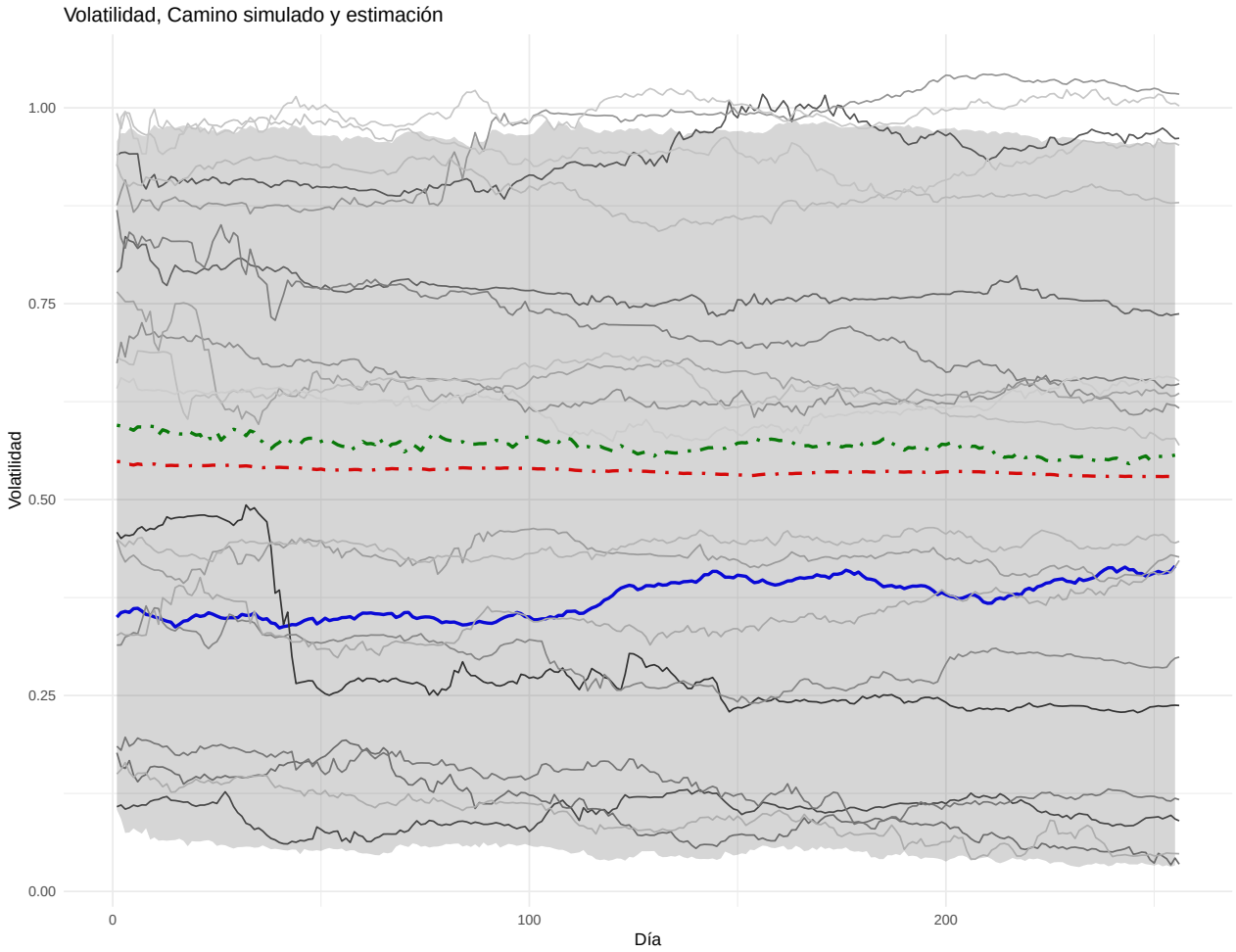


Figura 4.9: *Camino de la volatilidad simulada línea color azul. Línea discontinua roja valor promedio de la estimación, banda gris intervalo de confianza 90 %. Líneas grises son muestras de caminos estimados. Algoritmo SAMCMC.*

Para el algoritmo lo más difícil es reconstruir el camino de la variable latente, es decir el camino de la volatilidad. La Figura 4.9 replica la idea de la Figura anterior

con la adición de 20 caminos generados por el algoritmo, se aprecia que hay mucha dispersión en los caminos. La banda de confianza es grande y casi con el mismo ancho en todo el periodo.

A diferencia de los algoritmos LW y PMHM, este algoritmo no incorpora el paso de selección. En este paso se calcula la probabilidad $p(y_t^{ob}|\hat{x}_t^i, \hat{\boldsymbol{\theta}}^i)$ para cada partícula y se realiza un remuestreo con reemplazo, con lo cual las partículas más representativas quedan en la muestra del día t y se usan en la iteración siguiente $t+1$. Esta selección afecta directamente en la reconstrucción de los caminos, como se evidencia en las Figuras.

El algoritmo LW, como se puede ver en el Algoritmo 3.1, realiza un remuestreo adicional. Este remuestreo se realiza al inicio de la iteración con el propósito de escoger las partículas con mayor probabilidad de que cuando evolucionen en el tiempo $t+1$ sean más consistente con el valor observado y_{t+1} que las otras partículas. Esta característica junto con el factor del error o aleatoriedad produce que las estimaciones de los caminos sean mejores a pesar de la deficiente estimación de los parámetros.

Cuadro 4.13: *Raíz del error cuadrático medio (RECM) e Interval score (IS) con niveles 90 % y 80 % de las variables del modelo. Algoritmo SAMCMC.*

Variable	RECM	$IS_{90\,\%}$	$IS_{80\,\%}$
Volatilidad	0.31220	1.29910	0.92296
Log Precio	0.42067	0.89250	0.78645
α	0.01455	0.04371	0.03802
β	0.03224	0.09855	0.08763
H	0.08886	0.25959	0.20863
μ	0.00050	0.00140	0.00130

También hay que considerar que sobre esta variable latente recae toda la dispersión del sistema. Hay tres fuentes de dispersión (α, β, H), que son los parámetros de la

ecuación de estado.

Las medidas a usar para las comparaciones entre algoritmos se muestran en el Cuadro 4.13. El tiempo promedio de las 10 ejecuciones es de 35.80 horas.

4.3.1. Sensibilidad al periodo de quema en los MCMC's

Se ejecutó el algoritmo varias veces cambiando el periodo de quema con el fin de ver su comportamiento. Los periodos de quema que se probaron fueron: 100, 500, 1500, 1000 (escenario base). El número de partículas que se usó en cada escenario es de 130.

Cuadro 4.14: *Coeficiente de variación de los escenarios de sensibilidad al periodo de quema algoritmo SAMCMC.*

Quema	α	β	H	μ
100	43.07 %	43.87 %	19.84 %	49.45 %
500	47.22 %	39.73 %	14.16 %	35.34 %
1,000	17.04 %	29.22 %	7.78 %	31.34 %
1,500	25.42 %	31.12 %	11.22 %	25.40 %
2,000	29.09 %	21.25 %	13.92 %	36.90 %

Los Cuadros 4.14 y 4.15 muestran los resultados de los parámetros así como el tiempo de ejecución. Se nota que:

- El crecimiento del tiempo es lineal con respecto al periodo de quema.
- Los parámetros α y β no tienen tan clara su tendencia, pero en general su desviación estándar tiende a disminuir.
- El valor promedio del parámetro H tiende a disminuir con el aumento del periodo de quema y la menor desviación estándar se obtuvo con 1,000.
- El valor promedio del parámetro μ tiende a aumentar conforme aumenta el periodo de quema y su desviación estándar es muy simular entre los escenarios.

- La muestra más homogénea se logró con un periodo de quema de 1,000.

Cuadro 4.15: *Valores promedio y desviación estándar para diferentes escenarios de periodo de quema en los MCMC's algoritmo SAMCMC.*

Quema	Tiempo (Horas)		α	β	H	μ
100	3.73	mean	0.02626	0.04169	0.64018	0.00091
		sd	0.01131	0.01829	0.12701	0.00045
500	18.09	mean	0.02810	0.05178	0.62088	0.00116
		sd	0.01327	0.02057	0.08794	0.00041
1,000	35.66	mean	0.03299	0.09884	0.61025	0.00134
		sd	0.00562	0.02888	0.04750	0.00042
1,500	53.28	mean	0.03128	0.05354	0.55786	0.00189
		sd	0.00795	0.01666	0.06259	0.00048
2,000	71.33	mean	0.02025	0.05530	0.49280	0.00168
		sd	0.00589	0.01175	0.06862	0.00062
Real			**0.02733**	**0.07567**	**0.60000**	**0.00140**

4.3.2. Sensibilidad al periodo de quema y número de partículas

En estos escenarios se varió el número de partículas entre 130, 500 y 1000, con un periodo de quema de 100. Así como número de partículas 130, 150 y 1000 con periodo de quema de 1000.

Cuadro 4.16: *Coeficiente de variación de los escenarios de sensibilidad al periodo de quema y número de partículas. Algoritmo SAMCMC.*

Quema	Partículas	α	β	H	μ
100	130	43 %	44 %	20 %	49 %
100	500	48 %	49 %	23 %	45 %
100	1,000	44 %	43 %	21 %	50 %
1,000	130	17 %	29 %	8 %	31 %
1,000	150	23 %	24 %	12 %	29 %
1,000	1,000	32 %	28 %	17 %	28 %

Cuadro 4.17: *Valores promedio y desviación estándar para diferentes escenarios de periodo de quema y número de partículas en los MCMC's. Algoritmo SAMCMC.*

Quema	Partículas	Tiempo (Horas)		α	β	H	μ
100	130	3.73	mean	0.02626	0.04169	0.64018	0.00091
			sd	0.01131	0.01829	0.12701	0.00045
100	500	13.71	mean	0.02943	0.03461	0.63181	0.00129
			sd	0.01411	0.01683	0.14370	0.00058
100	1,000	27.33	mean	0.02533	0.03679	0.65284	0.00133
			sd	0.01122	0.01579	0.13501	0.00066
1,000	130	35.66	mean	0.03299	0.09884	0.61025	0.00134
			sd	0.00562	0.02888	0.04750	0.00042
1,000	150	40.13	mean	0.04528	0.08410	0.61254	0.00113
			sd	0.01048	0.02001	0.07613	0.00033
1,000	1,000	263.28	mean	0.03225	0.05652	0.57516	0.00139
			sd	0.01029	0.01593	0.09794	0.00039
	Real			**0.02733**	**0.07567**	**0.60000**	**0.00140**

En los Cuadros 4.17 y 4.16 se muestras los resultados de los escenarios. Podemos

observar que:

- El crecimiento del tiempo es lineal con respecto al número de partículas.
- Como era de esperar, el aumento en el periodo de quema produce muestras más homogéneas.
- Para los parámetros α y H con un periodo de quema de 1000, se observa que al aumentar el número de partículas las muestras tienden a aumentar la variabilidad.
- En el caso de β y μ con periodo de quema de 1000, el aumento en el número de partículas no muestra un tendencia clara.

4.4. Filtro de partículas Secuencial Aumentado MCMC con remuestreo

Dado que el algoritmo SAMCMC se implementó sin el paso de remuestreo y se obtuvo estimaciones buenas para los parámetros pero malas para la estimación de los caminos se procedió a implementar el algoritmo con el paso de remuestreo.

Un resumen de los valores estadísticos se muestran en el Cuadro 4.18. En la Figura 4.10 se presenta los histogramas de las muestras obtenidas.

El remuestreo provoca una reducción de elementos únicos en la muestra esto se puede apreciar al comparar la Figura 4.7 y 4.10.

En comparación con SAMCMC el agregar remuestreo desmejora la estimación de los parámetros, ver Cuadro 4.19.

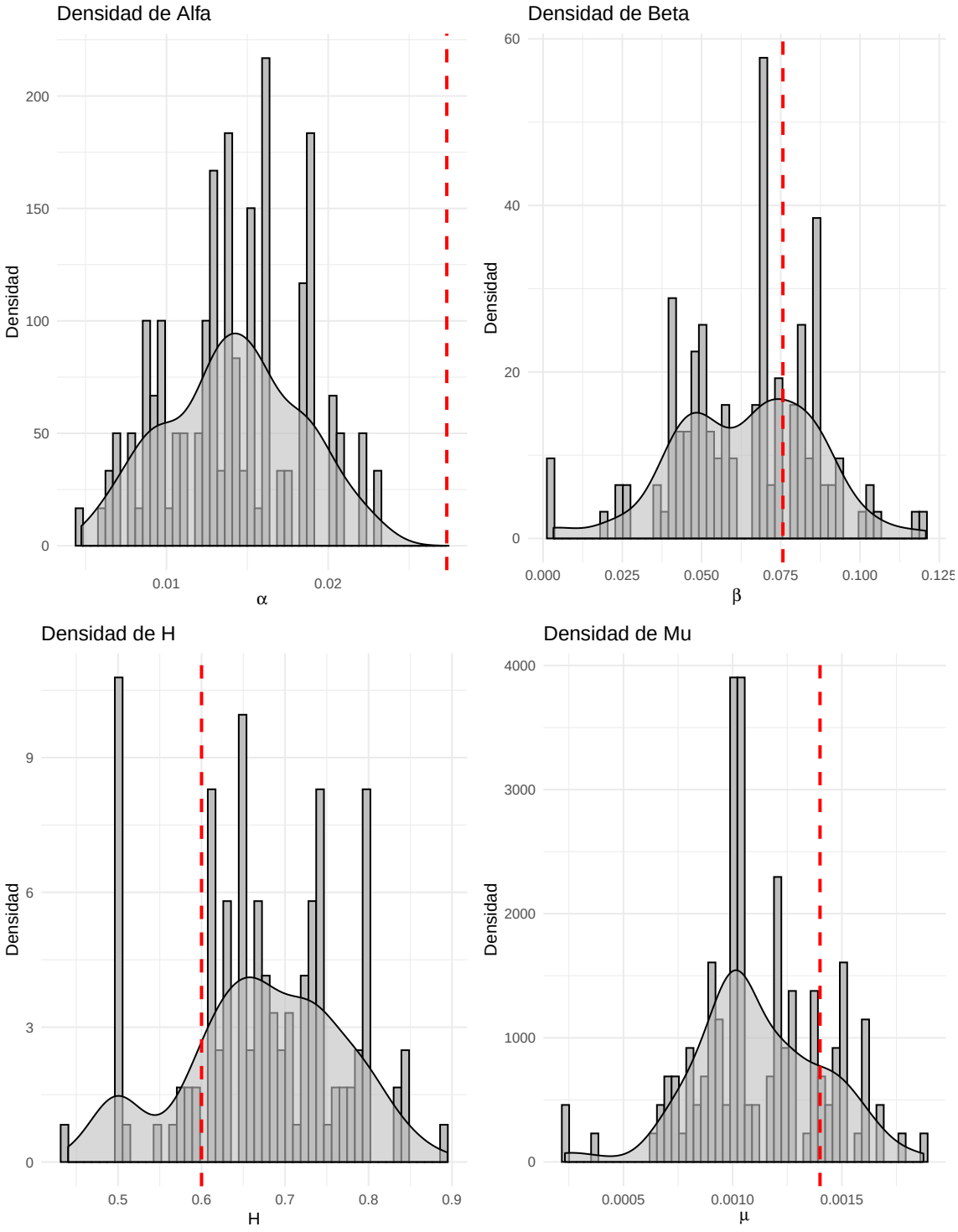


Figura 4.10: *Histogramas de las muestras obtenidas con SAMCMC con remuestreo. La línea discontinua de color rojo muestra el valor real. El área sombreada color gris corresponde a la estimación de la densidad por medio de un kernel gaussiano.*

Cuadro 4.18: *Resumen de valores estadísticos de las muestras obtenidas con SAMCMC con remuestreo.*

	α	β	H	μ
Min.:	0.004734	0.003275	0.4401	0.0002320
5 % Pe.:	0.007247	0.025687	0.4971	0.0007125
1st Qu.:	0.011274	0.048534	0.6171	0.0009292
Median:	0.014037	0.068791	0.6733	0.0010446
Mean:	0.014167	0.064153	0.6735	0.0011197
3rd Qu.:	0.017026	0.081621	0.7381	0.0013254
95 % Pe.:	0.020627	0.094687	0.7987	0.0016118
Max.:	0.023010	0.120786	0.8946	0.0018735

Cuadro 4.19: *Valor promedio y desviación estándar de la estimación de los parámetros junto con el valor real a estimar con SAMCMC con remuestreo.*

	α	β	H	μ
Mean	0.01417	0.06415	0.6735	0.00112
Sd	0.00412	0.02223	0.0949	0.00029
CV	0.29076	0.34653	0.1409	0.25893
Real	0.02733	0.07567	0.6000	0.00140

El beneficio que se obtiene por el remuestreo es en la estimación del camino del logaritmo del precio y el camino de la estimación de la volatilidad. Esto se puede ver en las Figuras 4.11 y 4.12.

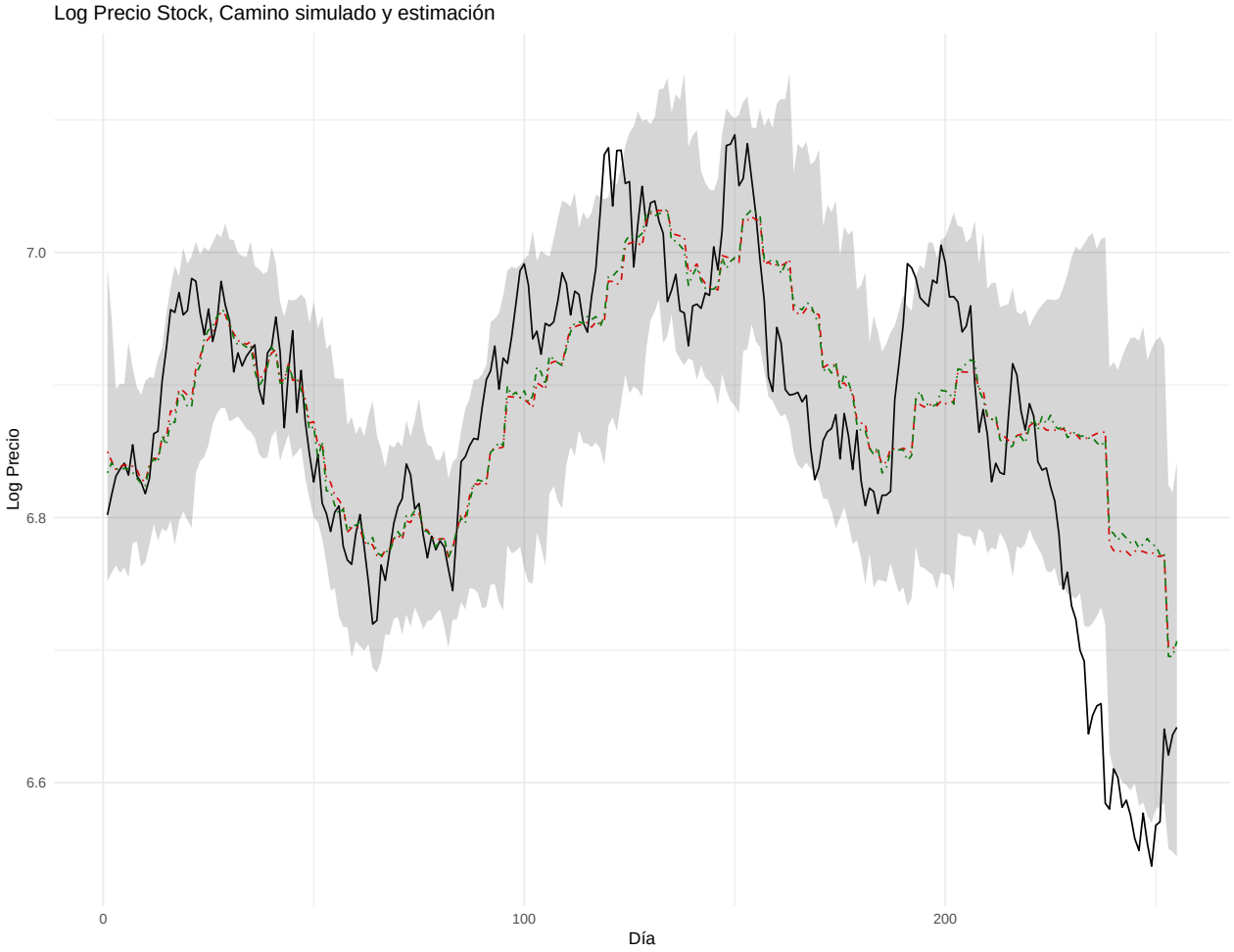


Figura 4.11: *Camino del Log Precio Stock simulado. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde es mediana, banda gris intervalo de confianza 90 %. Algoritmo SAMCMC con remuestreo.*

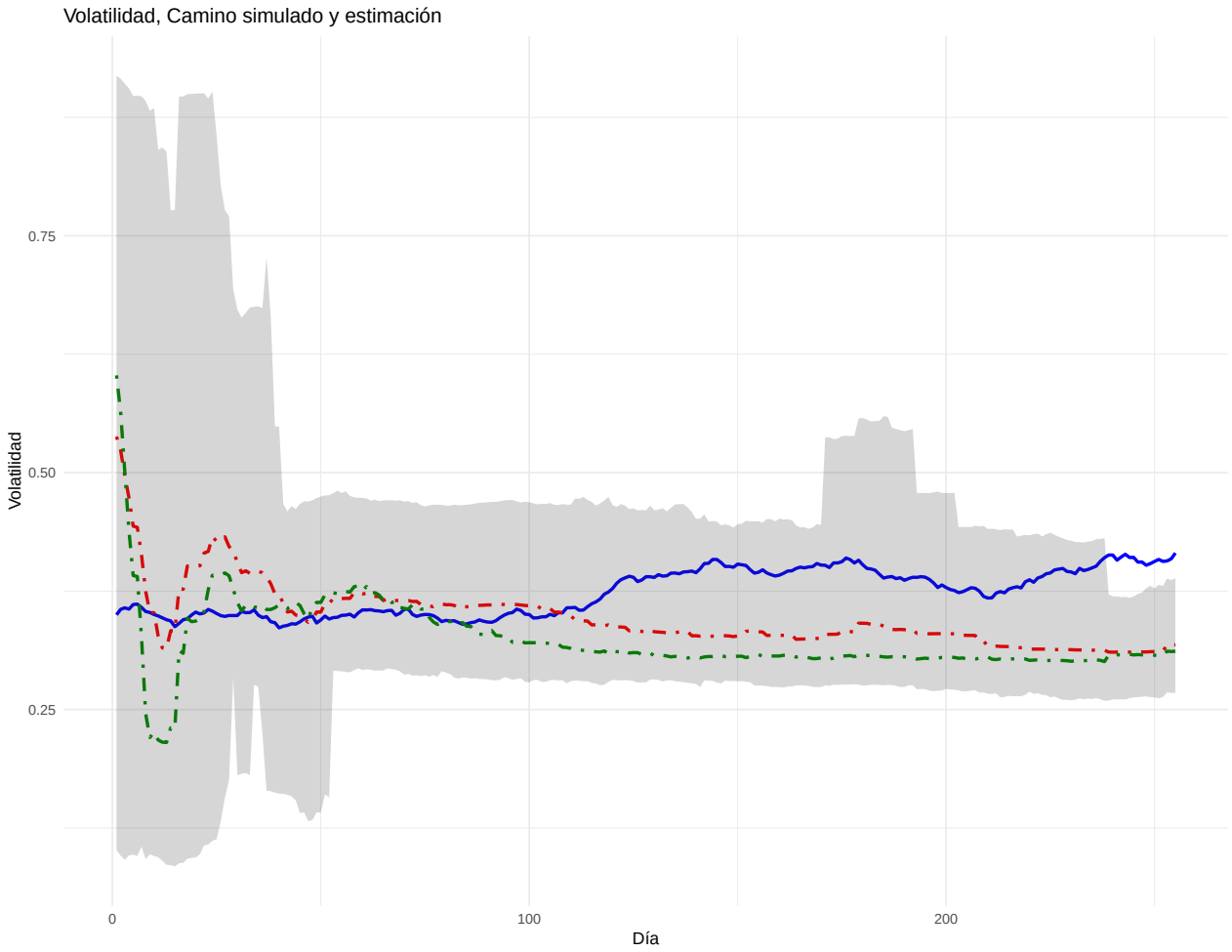


Figura 4.12: *Camino de la volatilidad simulada línea color azul. Línea discontinua roja valor promedio de la estimación, banda gris intervalo de confianza 90 %. Líneas grises son muestras de caminos estimados. Algoritmo SAMCMC con remuestreo.*

Las métricas respectivas para este algoritmo se muestran en el Cuadro 4.20. El tiempo promedio de las 10 ejecuciones es de 35.72 horas.

Cuadro 4.20: *Raíz del error cuadrático medio (RECM) e Interval score (IS) con niveles 90 % y 80 % de las variables del modelo. Algoritmo SAMCMC con remuestreo.*

Variable	RECM	$IS_{90\,\%}$	$IS_{80\,\%}$
Volatilidad	0.14114	0.39840	0.30789
Log Precio	0.09486	0.65224	0.51636
α	0.01287	0.10432	0.06461
β	0.03756	0.43383	0.25860
H	0.11488	0.28125	0.26501
μ	0.00059	0.00162	0.00140

4.5. Resumen de resultados

En esta sección se comparan el tiempo de ejecución, variabilidad, la raíz del error cuadrático medio y el score de intervalo para los diferentes algoritmos.

El algoritmo que dura menos es LW con 6.60 minutos, esto se debe a que el algoritmo sólo itera sobre el tiempo $t = 1, \ldots, 255$. El algoritmo PMMH tiene un tiempo de ejecución de 11.84 horas, cabe recalcar que este algoritmo ejecuta el filtro de partículas estándar 10,000 veces. El método SAMCMC tiene un tiempo de ejecución de 35.80 horas y SAMCMC con remuestreo un tiempo de 35.72 horas, son los más exigentes en cálculos. Estos últimos realizan en cada iteración t de la variable tiempo 130 MCMC, es decir una cadena para cada partícula, cada una con 1,000 iteraciones de periodo de quema.

En cuanto a la variabilidad de la estimación de los parámetros, el algoritmo de LW tiene los coeficientes de variación más bajas, pero la estimación que se obtiene es la más deficiente de todas. Con respecto a los otros algoritmos es SAMCMC el que tiene menor variabilidad. Por ejemplo el parámetro H en el algoritmo SAMCMC se obtuvo un coeficiente de variación de 0.07783, mientras que en los algoritmos PMMH y

SAMCMC con remuestreo prácticamente se duplico la variabilidad, pasando a 0.12649 y 0.1409 respectivamente.

Cuadro 4.21: *Comparación de la raíz del error cuadrático medio. Promedios sobre las 10 ejecuciones.*

	RECM			
	LW	PMMH	SAMCMC	SAMCMCR
Volatilidad	*0.1650*	0.2013	0.3122	**0.1411**
Log Precio	**0.0426**	0.1060	0.4207	*0.0949*
α	0.1080	0.0178	*0.0146*	**0.0129**
β	0.0682	0.0630	**0.0322**	*0.0376*
H	0.2789	0.1719	**0.0889**	*0.1149*
μ	0.1347	0.0008	**0.0005**	*0.0006*

Con respecto a la raíz del error cuadrático medio, ver Cuadro 4.21, vemos que la volatilidad se estimó mejor con SAMCMC con remuestreo y el Log precio se estimó mejor con LW pero como se indicó su estimación de parámetros es deficiente. La segunda mejor estimación es LW para la volatilidad y SAMCMC con remuestreo para el Log Precio. Por otra parte, es SAMCMC en general el que tiene mejor RECM en la estimación de los parámetros y SAMCMC con remuestreo el segundo mejor.

En los Cuadros 4.22 y 4.23 se ve que SAMCMC es el algoritmo que tiene mejores valores de $IS_{90\,\%}$ y $IS_{80\,\%}$ para los parámetros. Estos valores indican que su banda de confianza es mejor predictora en relación al resto de algoritmos. Y en general es SAMCMC con remuestreo el que tiene el segundo lugar en mejores valores en todos los variables.

En resumen, conforme se aumenta la complejidad de los algoritmos las estimaciones de los caminos de las variables Log precio y la volatilidad desmejoran, mientras que la estimación de los parámetros del modelo mejora. Es el algoritmo SAMCMC con

remuestreo el que tiene un balance de medidas buenas en todas las variables.

Cuadro 4.22: *Comparación del score de intervalo. Promedios sobre las 10 ejecuciones.*

	$IS_{90\,\%}$			
	LW	PMMH	SAMCMC	SAMCMCR
Volatilidad	**0.1464**	0.5510	1.2991	*0.3984*
Log Precio	1.4245	**0.4726**	0.8925	*0.6522*
α	2.0915	*0.0547*	**0.0437**	0.1043
β	1.2755	*0.1963*	**0.0986**	0.4338
H	5.5090	0.3839	**0.2596**	*0.2813*
μ	2.6058	0.0023	**0.0014**	*0.0016*

Cuadro 4.23: *Comparación del score de intervalo. Promedios sobre las 10 ejecuciones.*

	$IS_{80\,\%}$			
	LW	PMMH	SAMCMC	SAMCMCR
Volatilidad	**0.1116**	0.4165	0.9230	*0.3079*
Log Precio	0.8638	**0.3793**	0.7865	*0.5164*
α	1.0562	*0.0431*	**0.0380**	0.0646
β	0.6487	*0.1549*	**0.0876**	0.2586
H	2.7629	0.5294	**0.2086**	*0.2650*
μ	1.3139	0.0018	**0.0013**	*0.0014*

En conclusión, el mejor algoritmo para estimar los parámetros del modelo es SAMCMC, pues tiene los valores más bajos en RECM, IS y variabilidad. Pero es SAMCMC con remuestreo en general el mejor algoritmo porque logra estimar mejor los caminos y parámetros.

4.6. Aplicación con datos reales

En esta parte del estudio se tomó el mejor algoritmo, Secuencial Aumentado MCMC (SAMCMC) con remuestreo, y se aplicó en una serie corta y otra larga del índice S&P500. Se usó este índice porque el artículo [6] indica que hay evidencia de memoria larga, además porque usaron los datos de la serie corta para aplicar el procedimiento presentado en el artículo. Los datos se encuentran en el anexo A.2.

4.6.1. Datos de un mes

La serie corta del índice S&P500 corresponde a 22 observaciones diarias entre las fechas 2008-12-29 y 2009-01-29. El Cuadro 4.24 presenta los valores promedios estimados, 3 percentiles, la desviación estándar y el coeficiente de variación. El parámetro H presenta la variabilidad más baja al tener un coeficiente de variación de 0.20, mientras que los otros parámetros su variabilidad es mayor pero similar entre ellos.

Cuadro 4.24: *Valores estadísticos de los parámetros obtenidos con SAMCMC con remuestreo en datos de S&P500 del periodo 2008-12-29 al 2009-01-29. Los valores de Chronopoulou corresponden al estudio realizado en [6].*

	α	β	H	μ
mean	0.01896	0.04585	0.69483	0.00114
sd	0.00925	0.02118	0.13986	0.00047
CV	0.48787	0.46194	0.20129	0.41228
5 % Pc	0.00645	0.01448	0.43830	0.00039
50 % Pc	0.01966	0.04567	0.70971	0.00117
95 % Pc	0.03239	0.08142	0.94383	0.00186
Chronopoulou	0.05850	NA	0.53000	0.00540

Con esta misma serie de datos el artículo de [6] estimó valores para α, H y μ, los cuales se muestran en el Cuadro 4.24. A diferencia del artículo [6] en este trabajo se

estiman los parámetros de forma conjunta.

El tiempo que tomó la ejecución fue de 35.28 minutos.

La Figura 4.13 presenta el camino del logaritmo del índice S&P500 junto a la estimación que corresponde al valor promedio de las muestras obtenidas por el algoritmo para cada uno de los días. La banda gris muestra una zona de confianza de 90 %. Se tiene una buena estimación.

En la Figura 4.14 se presenta el camino de la volatilidad, construida con los promedios de las muestras diarias, además se presenta la mediana.

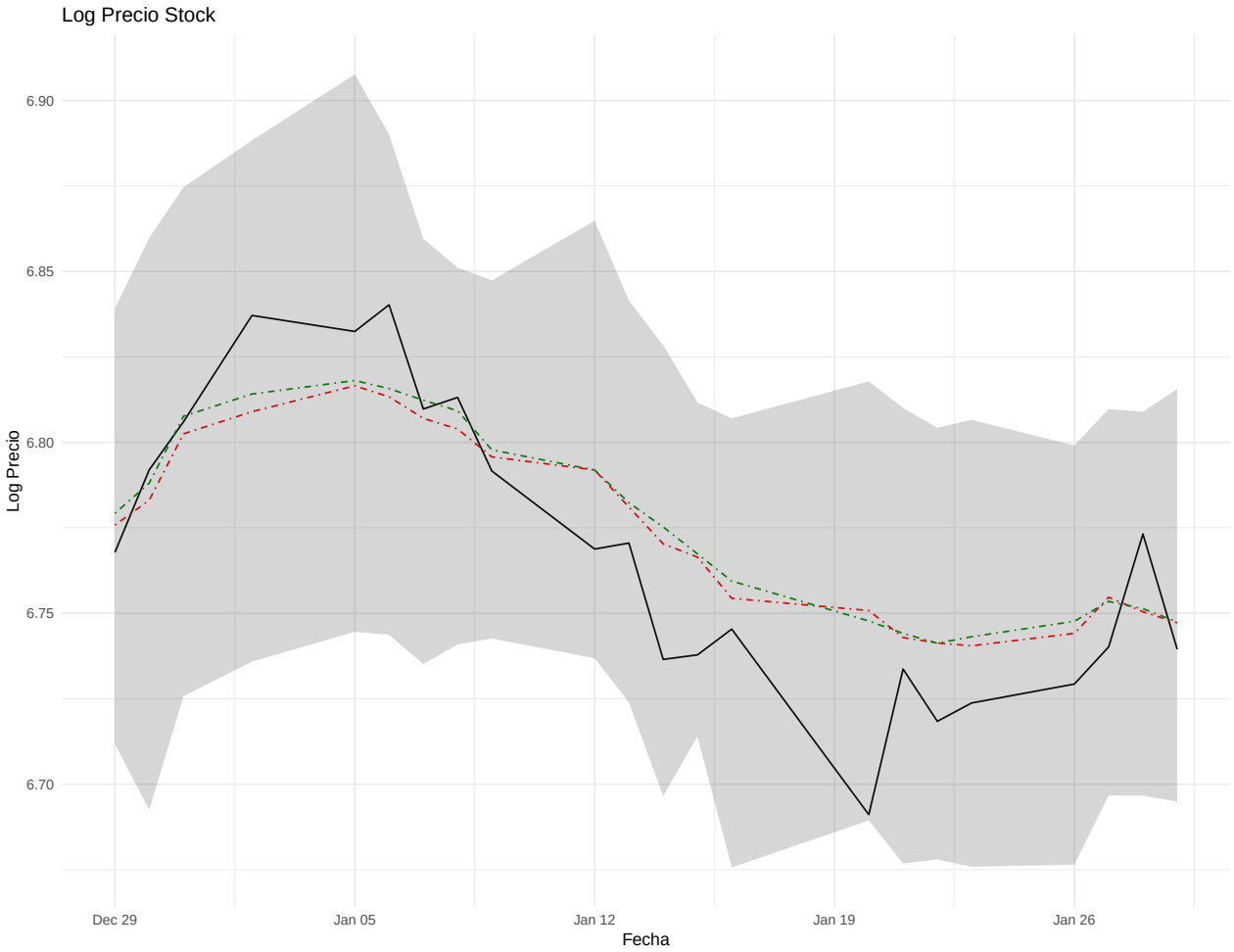


Figura 4.13: *Camino del Log S&P500 del periodo 2008-12-29 al 2009-01-29. Línea discontinua roja valor promedio de la estimación. La línea discontinua verde mediana, banda gris intervalo de confianza 90 %.*

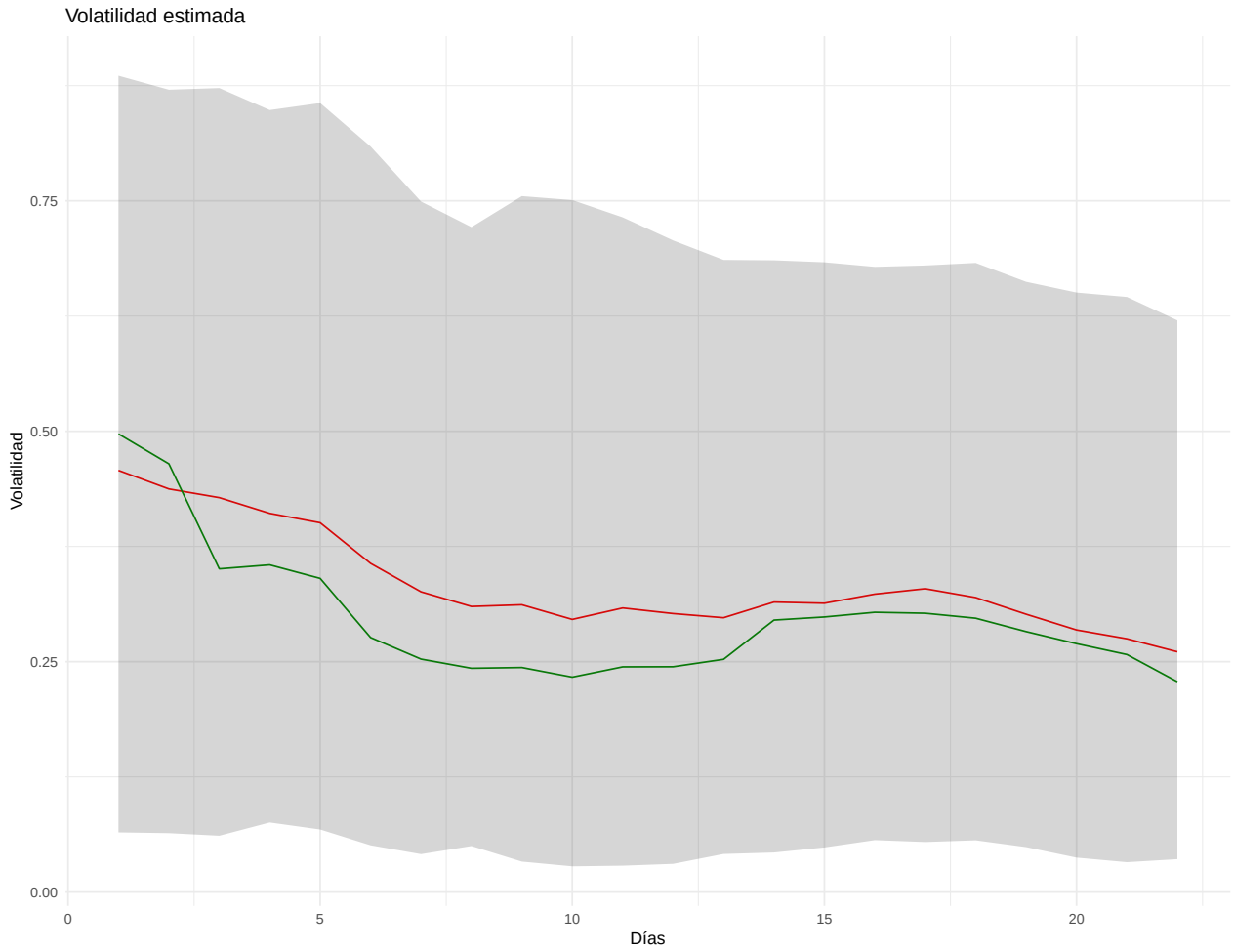


Figura 4.14: *Camino de la volatilidad estimada usando los datos de S&P500 del periodo 2008-12-29 al 2009-01-29. Línea discontinua roja valor promedio y la línea discontinua verde mediana.*

4.6.2. Datos de un año

Para la serie larga del índice S&P500 se tomó información entre las fechas 2008-12-29 y 2009-12-28, en total son 252 observaciones. El Cuadro 4.25 muestra los valores promedios estimados, 3 percentiles, la desviación estándar y el coeficiente de variación. Las desviaciones estándar disminuye: por ejemplo, el parámetro H tuvo una disminución de 64 % con respecto a la estimación usando la serie corta. Con respecto a la estimación el valor obtenido para el parámetro H es 0.70, aumentó levemente con res-

pecto a la estimación con los datos de la serie corta y con los dos ejercicios se evidencia la presencia de memoria larga. Por otro lado, al comparar el coeficiente de variación la variabilidad de las estimaciones se redujo considerablemente.

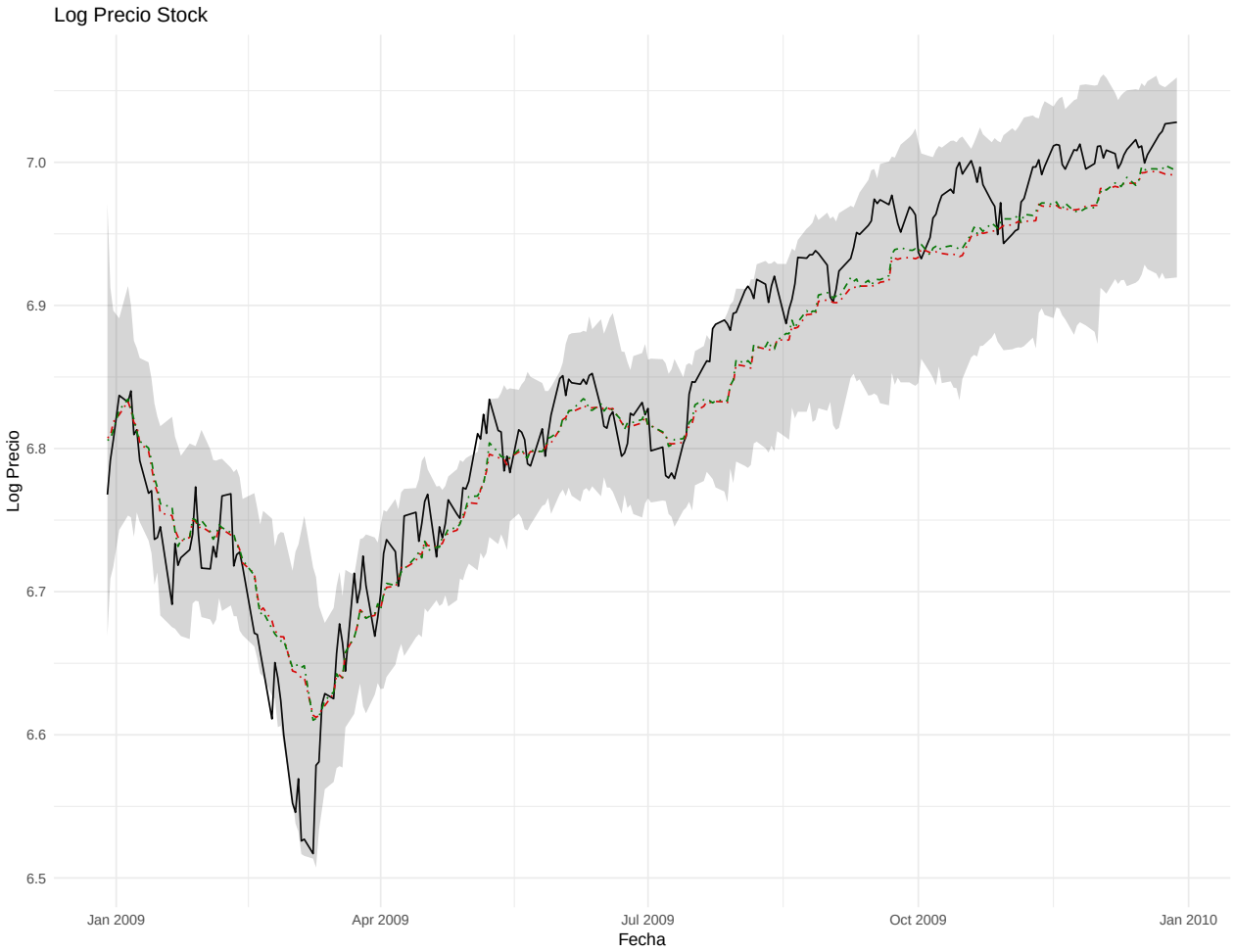


Figura 4.15: *Camino del Log S&P500 del periodo 2008-12-29 al 2009-12-28. Línea discontinua roja valor promedio de la estimación. Línea discontinua verde mediana, banda gris intervalo de confianza 90 %.*

Al igual que para la serie corta las Figuras 4.15 y 4.16 muestran la estimación del logaritmo del índice y la volatilidad. La estimación del camino del Log Precio es buena, su banda de confianza es angosta y el camino observado está en la banda.

Cuadro 4.25: *Valores estadísticos de los parámetros obtenidos con SAMCMC en datos de S&P500 del periodo 2008-12-29 al 2009-12-28.*

	α	β	H	μ
mean	0.03257	0.02463	0.70955	0.00118
sd	0.01047	0.00702	0.04979	0.00028
CV	0.32146	0.28502	0.07017	0.23729
5 % Pc	0.01691	0.01532	0.63779	0.00069
50 % Pc	0.03203	0.02399	0.71859	0.00116
95 % Pc	0.04793	0.03733	0.79704	0.00162

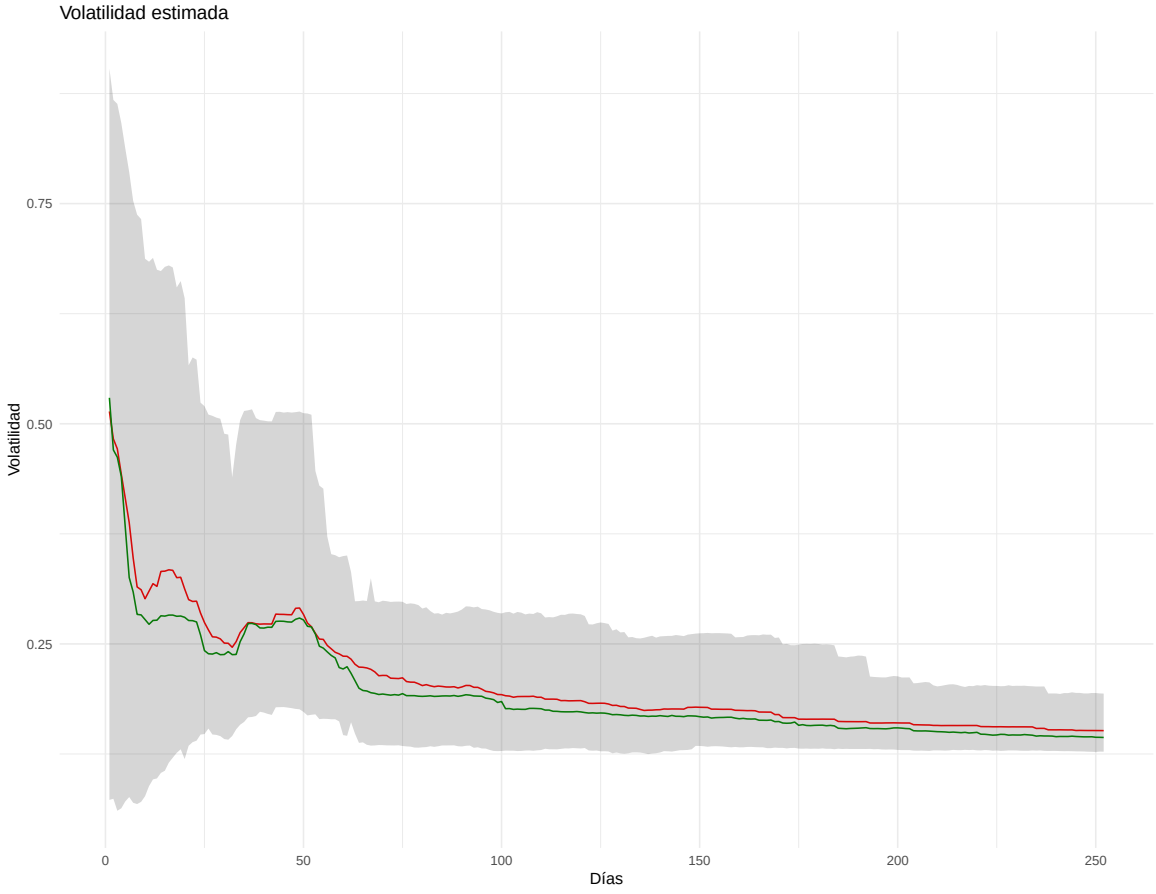


Figura 4.16: *Camino de la volatilidad estimada usando los datos de S&P500 del periodo 2008-12-29 al 2009-12-28. Línea rojo es el promedio y la línea verde es la mediana.*

CAPÍTULO 5

CONCLUSIONES GENERALES Y RECOMENDACIONES

En la sección 5.1 de este capítulo se comentan las conclusiones y en la sección 5.2 se presentan las recomendaciones de trabajos futuros.

5.1. Conclusiones

En este estudio se estimaron los parámetros de un modelo de volatilidad estocástica con memoria larga. Se usaron métodos de filtro de partículas y Cadenas de Markov vía Monte Carlo (MCMC por sus siglas en inglés). En la literatura este problema se había trabajado, pero la estimación de los parámetros no se había realizado de forma conjunta, es decir, el parámetro H en el proceso de memoria larga se había estimado de forma separada al resto de parámetros.

El problema es complejo, además de la estimación de los parámetros se debe estimar el camino de la variable observada y el camino de la variable latente. Esto nos lleva a un espacio de búsqueda muy grande y que además va aumentando conforme transcurre la variable del tiempo.

Hay varias variables de configuración que se deben definir para poder ejecutar los

algoritmos, como son el número de partículas, el tamaño de muestra para los MCMC, el periodo de quema. En el caso del número de partículas se tuvo que escoger un valor tomando en cuenta su efecto directo en el tiempo de duración de los algoritmos, por lo que se trató de llegar a un balance entre el número de partículas y el tiempo.

Se tuvo una consideración importante con respecto al método de filtro de partículas. Este método asume que los procesos del modelo de espacio estado cumplen con la propiedad de Markov, supuesto que no se cumple para la variable latente en este trabajo. Se realizó una revisión de la literatura y se encontró un artículo en que aplicaban filtro de partículas a procesos no Markov y en el argumentan que de igual forma funcionaban [19].

Se simularon datos con el propósito de poder comparar los 4 algoritmos considerados. Conociendo el valor exacto de los parámetros y los caminos, se aplicaron medidas para valorar la efectividad y desempeño de los algoritmos, por lo que se pudo determinar cual es el que produce mejores estimaciones.

El primer algoritmo que se consideró fue el filtro de partículas de Liu-West (LW), este método se visualizó como un punto de referencia, pues es un algoritmo muy conocido en el ambiente de filtros de partículas. El algoritmo es una modificación del filtro de partículas auxiliar, el cambio principal es que se agrega una dinámica artificial a la evolución de los parámetros. Este algoritmo estimó muy bien el camino de la variable observada, produjo una estimación aceptable de la variable latente, pero la estimación de los parámetros del modelo fue muy mala. Se detectó que el buen desempeño en la estimación de los caminos se debe a que el método realiza una escogencia de las mejores partículas al inicio y final de cada iteración. Esta escogencia o remuestreo se basa en la información observada, de ahí su efectividad en la estimación de la variable observada.

El segundo algoritmo que se programó fue el filtro de partículas marginal Metropolis-Hastings (PMMH). El algoritmo es un Metropolis-Hastings en el cual se usa el filtro de partículas para determinar la probabilidad de aceptación del nuevo candidato $\boldsymbol{\theta}$ a

formar parte de la muestra. La estimación de los parámetros es mejor que la obtenida por LW, pero la estimación de los caminos desmejora considerablemente. La principal desventaja de este método es la gran influencia de la distribución a priori, que para este estudio no se contaba con información previa para poder determinarla.

El filtro de partículas secuencial aumentado MCMC (SAMCMC) fue la tercera opción. Este algoritmo es mucho más complejo que los dos primeros. Genera en cada iteración del tiempo un MCMC para cada una de las partículas para obtener el valor siguiente después de un periodo de quema. Con estos valores se construye la muestra en la iteración t. El método permite realizar una programación en paralelo para mejorar el tiempo de ejecución y aún así es el algoritmo que más tiempo toma, alrededor de 35 horas. Con este algoritmo se obtuvieron los mejores resultados en la estimación de los parámetros: los valores reales de los parámetros se ubican en regiones de gran probabilidad de las distribuciones empíricas a posterior. Pese a lo anterior, las estimaciones obtenidas para los caminos son deficientes, además de tener bandas de confianza muy anchas.

Teniendo que SAMCMC se había implementado sin remuestreo y que su estimación de los caminos fue mala, se decidió implementar SAMCMC con remuestreo. Se logró una mejora considerablemente en las estimaciones de los caminos a tal punto que compiten con los de LW. Pese a lo anterior, se sacrificó en la estimación de los parámetros, por lo que los resultados no son tan buenos como SAMCMC. Según las medidas usadas, este algoritmo es en general el segundo mejor en todas las estimaciones. Con respecto al tiempo de ejecución se demora aproximadamente lo mismo que SAMCMC.

Habiendo determinado que el mejor algoritmo es SAMCMC con remuestreo se procedió a aplicarlo en datos reales. Se tomó datos del índice Standard & Poor's 500 para un mes y de un año. Las dos estimaciones dan evidencia de la existencia de memoria larga, en los parámetros β y H hay mayor desviación en la estimación con datos de un mes, caso contrario con α y μ. El camino estimado de la variable observada es muy

bueno y la estimación del camino de la volatilidad disminuye y se estabiliza por 15 %. No hay mayor diferencia en usar el promedio o la mediana como estimador.

En conclusión, se obtuvo un algoritmo que logra estimar de forma conjunta los parámetros del modelo de volatilidad estocástica con memoria larga, que además estima satisfactoriamente los caminos del modelo de espacio estado. Se puede decir que la desventaja que se tiene es el tiempo de ejecución, que se considerablemente grande.

5.2. Recomendaciones de trabajos futuros

Posibles recomendaciones de mejoras para trabajos futuros se presentan a continuación.

- Para mejorar los tiempos de ejecución se puede usar otro método para la estimación del movimiento browniano fraccionario estándar. El método que se usó en teoría es más exacto, pero se han desarrollado métodos más veloces. Además se puede usar algún paquete que ya tenga implementado el método para no tener que programarlo.

- Se puede revisar la programación del algoritmo SAMCMC con remuestreo para mejorarlo en aspectos de velocidad. Explorar la posibilidad de implementar otras partes del código en paralelo.

- Se puede realizar un estudio más profundo en la obtención de las distribuciones iniciales y la distribución propuesta.

- Otro aspecto que se puede profundizar es la medida usada para obtener los pesos.

- Un posible trabajo es investigar e implementar otros métodos para realizar el remuestreo.

Apéndices

APÉNDICE $\mathcal{A}$

DATOS

A.1. Datos simulados

Cuadro A.1: *Detalle de los valores simulados.*

Día	Ln Precio	Volatilidad	Día	Ln Precio	Volatilidad	Día	Ln Precio	Volatilidad
1	6.8020	0.3500	86	6.8459	0.3417	171	6.8582	0.4023
2	6.8172	0.3558	87	6.8541	0.3424	172	6.8648	0.3999
3	6.8311	0.3573	88	6.8597	0.3446	173	6.8672	0.4048
4	6.8366	0.3560	89	6.8589	0.3432	174	6.8777	0.4047
5	6.8410	0.3608	90	6.8831	0.3424	175	6.8443	0.4060
6	6.8320	0.3613	91	6.9040	0.3420	176	6.8785	0.4099
7	6.8549	0.3578	92	6.9106	0.3438	177	6.8618	0.4082
8	6.8315	0.3535	93	6.9293	0.3470	178	6.8363	0.4046
9	6.8266	0.3525	94	6.8968	0.3496	179	6.8658	0.4076
10	6.8181	0.3503	95	6.9203	0.3516	180	6.8278	0.4028
11	6.8298	0.3488	96	6.9163	0.3524	181	6.8090	0.3989
12	6.8633	0.3468	97	6.9362	0.3561	182	6.8222	0.3984

13	6.8651	0.3448	98	6.9608	0.3545	183	6.8196	0.3970
14	6.9031	0.3436	99	6.9862	0.3506	184	6.8030	0.3928
15	6.9277	0.3374	100	6.9913	0.3503	185	6.8168	0.3888
16	6.9569	0.3409	101	6.9752	0.3466	186	6.8170	0.3898
17	6.9547	0.3449	102	6.9347	0.3468	187	6.8198	0.3904
18	6.9697	0.3458	103	6.9406	0.3478	188	6.8891	0.3882
19	6.9529	0.3497	104	6.9232	0.3479	189	6.9159	0.3891
20	6.9561	0.3529	105	6.9465	0.3505	190	6.9455	0.3864
21	6.9804	0.3509	106	6.9446	0.3490	191	6.9916	0.3882
22	6.9783	0.3519	107	6.9476	0.3522	192	6.9884	0.3896
23	6.9544	0.3557	108	6.9638	0.3520	193	6.9807	0.3895
24	6.9379	0.3538	109	6.9846	0.3574	194	6.9659	0.3902
25	6.9572	0.3514	110	6.9767	0.3574	195	6.9624	0.3897
26	6.9329	0.3492	111	6.9529	0.3578	196	6.9594	0.3873
27	6.9470	0.3483	112	6.9708	0.3547	197	6.9791	0.3836
28	6.9781	0.3493	113	6.9683	0.3550	198	6.9768	0.3788
29	6.9602	0.3492	114	6.9471	0.3586	199	7.0054	0.3814
30	6.9487	0.3492	115	6.9398	0.3617	200	6.9922	0.3787
31	6.9099	0.3535	116	6.9669	0.3636	201	6.9664	0.3767
32	6.9242	0.3523	117	6.9877	0.3666	202	6.9667	0.3756
33	6.9144	0.3524	118	7.0285	0.3717	203	6.9626	0.3734
34	6.9216	0.3552	119	7.0736	0.3740	204	6.9400	0.3744
35	6.9261	0.3497	120	7.0790	0.3777	205	6.9445	0.3764
36	6.9303	0.3470	121	7.0350	0.3833	206	6.9598	0.3786
37	6.8969	0.3481	122	7.0769	0.3871	207	6.9015	0.3780
38	6.8856	0.3427	123	7.0771	0.3890	208	6.8643	0.3749
39	6.9240	0.3417	124	7.0520	0.3907	209	6.8815	0.3694
40	6.9289	0.3362	125	7.0534	0.3896	210	6.8627	0.3677
41	6.9513	0.3380	126	6.9891	0.3850	211	6.8270	0.3679

42	6.9254	0.3388	127	7.0219	0.3867	212	6.8410	0.3729
43	6.8678	0.3405	128	7.0499	0.3902	213	6.8339	0.3745
44	6.9104	0.3402	129	7.0199	0.3902	214	6.8326	0.3731
45	6.9409	0.3426	130	7.0373	0.3894	215	6.8761	0.3771
46	6.8793	0.3457	131	7.0389	0.3924	216	6.9160	0.3785
47	6.9110	0.3475	132	7.0236	0.3908	217	6.9076	0.3796
48	6.8722	0.3485	133	7.0143	0.3912	218	6.8802	0.3784
49	6.8476	0.3413	134	6.9629	0.3940	219	6.8659	0.3848
50	6.8269	0.3446	135	6.9716	0.3941	220	6.8859	0.3870
51	6.8477	0.3487	136	6.9835	0.3933	221	6.8765	0.3843
52	6.8110	0.3455	137	6.9560	0.3952	222	6.8421	0.3889
53	6.8031	0.3471	138	6.9544	0.3955	223	6.8358	0.3903
54	6.7896	0.3474	139	6.9295	0.3962	224	6.8374	0.3933
55	6.8041	0.3494	140	6.9596	0.3944	225	6.8234	0.3942
56	6.8088	0.3497	141	6.9610	0.3985	226	6.8126	0.3974
57	6.7784	0.3507	142	6.9578	0.4041	227	6.7875	0.3982
58	6.7679	0.3476	143	6.9695	0.4043	228	6.7461	0.3989
59	6.7647	0.3521	144	6.9676	0.4084	229	6.7588	0.3957
60	6.7880	0.3552	145	7.0042	0.4084	230	6.7333	0.3952
61	6.8023	0.3550	146	6.9869	0.4052	231	6.7233	0.3933
62	6.7775	0.3554	147	7.0166	0.4009	232	6.6999	0.3990
63	6.7509	0.3544	148	7.0806	0.4010	233	6.6917	0.3966
64	6.7197	0.3540	149	7.0817	0.4001	234	6.6369	0.3978
65	6.7225	0.3533	150	7.0886	0.4035	235	6.6509	0.3995
66	6.7643	0.3544	151	7.0504	0.4028	236	6.6579	0.4013
67	6.7526	0.3547	152	7.0557	0.4020	237	6.6597	0.4064
68	6.7738	0.3496	153	7.0820	0.3976	238	6.5846	0.4103
69	6.7955	0.3516	154	7.0555	0.3940	239	6.5801	0.4131
70	6.8080	0.3558	155	7.0290	0.3949	240	6.6105	0.4130

71	6.8142	0.3561	156	6.9942	0.3975	241	6.6040	0.4076
72	6.8404	0.3504	157	6.9638	0.3941	242	6.5815	0.4110
73	6.8323	0.3483	158	6.9059	0.3924	243	6.5869	0.4139
74	6.8064	0.3497	159	6.8951	0.3909	244	6.5755	0.4104
75	6.8107	0.3503	160	6.9435	0.3919	245	6.5580	0.4106
76	6.7871	0.3502	161	6.9317	0.3938	246	6.5490	0.4056
77	6.7697	0.3491	162	6.8966	0.3962	247	6.5771	0.4057
78	6.7860	0.3465	163	6.8924	0.3964	248	6.5542	0.4023
79	6.7757	0.3428	164	6.8930	0.3989	249	6.5372	0.4041
80	6.7825	0.3441	165	6.8944	0.4003	250	6.5679	0.4063
81	6.7778	0.3434	166	6.8873	0.3995	251	6.5706	0.4084
82	6.7610	0.3437	167	6.8922	0.4005	252	6.6404	0.4063
83	6.7450	0.3419	168	6.8532	0.4008	253	6.6208	0.4068
84	6.7915	0.3400	169	6.8285	0.4041	254	6.6362	0.4090
85	6.8422	0.3404	170	6.8377	0.4025	255	6.6419	0.4152

A.2. Datos Reales

Los datos se obtuvieron de Yahoo Finance. Los datos se pueden descargar del siguiente link: `https://finance.yahoo.com/quote/%5EGSPC/history?period1=1230508800&period2=1261958400&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true`

Cuadro A.2: *Información histórica del índice S&P500.*

Date	Close	Date	Close	Date	Close	Date	Close
2008-12-29	869.4200	2009-03-31	797.8700	2009-06-30	919.3200	2009-09-29	1060.6100
2008-12-30	890.6400	2009-04-01	811.0800	2009-07-01	923.3300	2009-09-30	1057.0800

2008-12-31	903.2500	2009-04-02	834.3800	2009-07-02	896.4200	2009-10-01	1029.8500
2009-01-02	931.8000	2009-04-03	842.5000	2009-07-06	898.7200	2009-10-02	1025.2100
2009-01-05	927.4500	2009-04-06	835.4800	2009-07-07	881.0300	2009-10-05	1040.4600
2009-01-06	934.7000	2009-04-07	815.5500	2009-07-08	879.5600	2009-10-06	1054.7200
2009-01-07	906.6500	2009-04-08	825.1600	2009-07-09	882.6800	2009-10-07	1057.5800
2009-01-08	909.7300	2009-04-09	856.5600	2009-07-10	879.1300	2009-10-08	1065.4800
2009-01-09	890.3500	2009-04-13	858.7300	2009-07-13	901.0500	2009-10-09	1071.4900
2009-01-12	870.2600	2009-04-14	841.5000	2009-07-14	905.8400	2009-10-12	1076.1899
2009-01-13	871.7900	2009-04-15	852.0600	2009-07-15	932.6800	2009-10-13	1073.1899
2009-01-14	842.6200	2009-04-16	865.3000	2009-07-16	940.7400	2009-10-14	1092.0200
2009-01-15	843.7400	2009-04-17	869.6000	2009-07-17	940.3800	2009-10-15	1096.5601
2009-01-16	850.1200	2009-04-20	832.3900	2009-07-20	951.1300	2009-10-16	1087.6801
2009-01-20	805.2200	2009-04-21	850.0800	2009-07-21	954.5800	2009-10-19	1097.9100
2009-01-21	840.2400	2009-04-22	843.5500	2009-07-22	954.0700	2009-10-20	1091.0601
2009-01-22	827.5000	2009-04-23	851.9200	2009-07-23	976.2900	2009-10-21	1081.4000
2009-01-23	831.9500	2009-04-24	866.2300	2009-07-24	979.2600	2009-10-22	1092.9100
2009-01-26	836.5700	2009-04-27	857.5100	2009-07-27	982.1800	2009-10-23	1079.6000
2009-01-27	845.7100	2009-04-28	855.1600	2009-07-28	979.6200	2009-10-26	1066.9500
2009-01-28	874.0900	2009-04-29	873.6400	2009-07-29	975.1500	2009-10-27	1063.4100
2009-01-29	845.1400	2009-04-30	872.8100	2009-07-30	986.7500	2009-10-28	1042.6300
2009-01-30	825.8800	2009-05-01	877.5200	2009-07-31	987.4800	2009-10-29	1066.1100
2009-02-02	825.4400	2009-05-04	907.2400	2009-08-03	1002.6300	2009-10-30	1036.1899
2009-02-03	838.5100	2009-05-05	903.8000	2009-08-04	1005.6500	2009-11-02	1042.8800
2009-02-04	832.2300	2009-05-06	919.5300	2009-08-05	1002.7200	2009-11-03	1045.4100
2009-02-05	845.8500	2009-05-07	907.3900	2009-08-06	997.0800	2009-11-04	1046.5000
2009-02-06	868.6000	2009-05-08	929.2300	2009-08-07	1010.4800	2009-11-05	1066.6300
2009-02-09	869.8900	2009-05-11	909.2400	2009-08-10	1007.1000	2009-11-06	1069.3000
2009-02-10	827.1600	2009-05-12	908.3500	2009-08-11	994.3500	2009-11-09	1093.0800
2009-02-11	833.7400	2009-05-13	883.9200	2009-08-12	1005.8100	2009-11-10	1093.0100

2009-02-12	835.1900	2009-05-14	893.0700	2009-08-13	1012.7300	2009-11-11	1098.5100
2009-02-13	826.8400	2009-05-15	882.8800	2009-08-14	1004.0900	2009-11-12	1087.2400
2009-02-17	789.1700	2009-05-18	909.7100	2009-08-17	979.7300	2009-11-13	1093.4800
2009-02-18	788.4200	2009-05-19	908.1300	2009-08-18	989.6700	2009-11-16	1109.3000
2009-02-19	778.9400	2009-05-20	903.4700	2009-08-19	996.4600	2009-11-17	1110.3199
2009-02-20	770.0500	2009-05-21	888.3300	2009-08-20	1007.3700	2009-11-18	1109.8000
2009-02-23	743.3300	2009-05-22	887.0000	2009-08-21	1026.1300	2009-11-19	1094.9000
2009-02-24	773.1400	2009-05-26	910.3300	2009-08-24	1025.5699	2009-11-20	1091.3800
2009-02-25	764.9000	2009-05-27	893.0600	2009-08-25	1028.0000	2009-11-23	1106.2400
2009-02-26	752.8300	2009-05-28	906.8300	2009-08-26	1028.1200	2009-11-24	1105.6500
2009-02-27	735.0900	2009-05-29	919.1400	2009-08-27	1030.9800	2009-11-25	1110.6300
2009-03-02	700.8200	2009-06-01	942.8700	2009-08-28	1028.9301	2009-11-27	1091.4900
2009-03-03	696.3300	2009-06-02	944.7400	2009-08-31	1020.6200	2009-11-30	1095.6300
2009-03-04	712.8700	2009-06-03	931.7600	2009-09-01	998.0400	2009-12-01	1108.8600
2009-03-05	682.5500	2009-06-04	942.4600	2009-09-02	994.7500	2009-12-02	1109.2400
2009-03-06	683.3800	2009-06-05	940.0900	2009-09-03	1003.2400	2009-12-03	1099.9200
2009-03-09	676.5300	2009-06-08	939.1400	2009-09-04	1016.4000	2009-12-04	1105.9800
2009-03-10	719.6000	2009-06-09	942.4300	2009-09-08	1025.3900	2009-12-07	1103.2500
2009-03-11	721.3600	2009-06-10	939.1500	2009-09-09	1033.3700	2009-12-08	1091.9399
2009-03-12	750.7400	2009-06-11	944.8900	2009-09-10	1044.1400	2009-12-09	1095.9500
2009-03-13	756.5500	2009-06-12	946.2100	2009-09-11	1042.7300	2009-12-10	1102.3500
2009-03-16	753.8900	2009-06-15	923.7200	2009-09-14	1049.3400	2009-12-11	1106.4100
2009-03-17	778.1200	2009-06-16	911.9700	2009-09-15	1052.6300	2009-12-14	1114.1100
2009-03-18	794.3500	2009-06-17	910.7100	2009-09-16	1068.7600	2009-12-15	1107.9301
2009-03-19	784.0400	2009-06-18	918.3700	2009-09-17	1065.4900	2009-12-16	1109.1801
2009-03-20	768.5400	2009-06-19	921.2300	2009-09-18	1068.3000	2009-12-17	1096.0800
2009-03-23	822.9200	2009-06-22	893.0400	2009-09-21	1064.6600	2009-12-18	1102.4700
2009-03-24	806.1200	2009-06-23	895.1000	2009-09-22	1071.6600	2009-12-21	1114.0500
2009-03-25	813.8800	2009-06-24	900.9400	2009-09-23	1060.8700	2009-12-22	1118.0200

2009-03-26	832.8600	2009-06-25	920.2600	2009-09-24	1050.7800	2009-12-23	1120.5900
2009-03-27	815.9400	2009-06-26	918.9000	2009-09-25	1044.3800	2009-12-24	1126.4800
2009-03-30	787.5300	2009-06-29	927.2300	2009-09-28	1062.9800	2009-12-28	1127.7800

APÉNDICE $\mathcal{B}$

CÓDIGOS

En este apéndice se presentan los códigos desarrollados para realizar los cálculos.

Las siguientes variables son los parámetros generales del proceso simulado.

```
H <- 0.6
alpha <- 0.02733
m <- 0
beta <- 0.07567
mu <- 0.0014
N  <-  255
n <-  130
dt <- 1/N
```

Función que genera los datos simulados.

```
generaDatosSim <- function(x0 = 0.35, y0 = 6.802, semilla = 9) {

  x     <- NULL
  y     <- NULL
  y[1]  <- y0
  x[1]  <- x0
  dt <- 1/N

  set.seed(semilla)

```

```
  dBH <- diff(cholFBM(N, H, 1))

  for(t in 1:(N-1)){
    x[t+1] <- x[t] + alpha*(m - x[t])*dt + beta*dBH[t]
    y[t+1] <- y[t] + (mu-(x[t]^2)/2)*dt + x[t]*rnorm(1)*sqrt(dt)
  }

  return(list(y = y, x = x))
}
```

Conjunto de funciones usadas en los algoritmos.

```
propagaX <- function(x, alpha, m, beta, dBH) {
  return(x + alpha*(m - x)*dt + beta*dBH)
}

propagaXsinError <- function(x, alpha, m) {
  return(x + alpha*(m - x)*dt)
}

propagaY <- function(y, x, mu){
  n <- length(x)
  return(y + (mu-(x^2)/2)*dt + x*rnorm(n)*sqrt(dt))
}

propagaY2 <- function(y, x, mu){
  return(y + (mu-(x^2)/2)*dt + x*rnorm(1)*sqrt(dt))
}

propagaYsinError <- function(y, x, mu){
  return(y + (mu-(x^2)/2)*dt)
}

calculaVerosimilitud <- function(w){
  sum(w) / length(w)
}

discProb <- function(X.sim, x.real){
  n <- length(X.sim)
  phi <- function(x, n){
    n^(1/3)*exp(-2*abs(x*n^(1/3)))
  }
  phiValor <- phi(X.sim - x.real, n)
```

```
  return(phiValor/sum(phiValor))
}

discProb2 <- function(X.sim, x.real){
  n <- length(X.sim)
  phi <- function(x, n){
    n^(1/3)*exp(-2*abs(x*n^(1/3)))
  }
  phiValor <- phi(X.sim - x.real, n)

  return(phiValor)
}
```

Función que realiza un remuestreo.

```
sysResampling <- function(pesos) {
  "Algoritmo 2 Jacob pag 12"
  N <- length(pesos)
  resampling <- rep(0, N)

  u <- runif(1)
  pesos <- pesos / sum(pesos)

  uu <- u / N
  j <- 1
  sw <- pesos[1]
  for (k in 1:N) {
    while (sw < uu) {
      j <- j + 1
      sw <- sw + pesos[j]
    }
    resampling[k] <- j
    uu <- uu + 1/N
  }

  return(resampling)
}
```

Función para calcular la medida intervalo de score.

```
intervalScore <- function(x, a, muestra) {
  l = quantile(muestra, probs = c(a/2), names = FALSE)
  u = quantile(muestra, probs = c(1-a/2), names = FALSE)
```

```
  ind1 = ifelse(x < l, 1, 0)
  ind2 = ifelse(x > u, 1, 0)
  s = (u-l) + 2/a * (l-x)*ind1 + 2/a * (x-u)*ind2
  return(s)
}
```

Función para calcular el camino del proceso browniano fraccionario estándar usando el método de Choleski.

```
cholFBM <- function(n, H){
  # tomado de "SIMULATION AND IDENTIFICATION OF THE
  # FRACTIONAL BROWNIAN MOTION: A BIBLIOGRAPHICAL AND COMPARATIVE STUDY
  # COEURJOLLY Jean-Francois

  H2 <- 2 * H
  matcov <- matrix(0, n - 1, n - 1)

  for(i in (1:(n - 1))){
    for(j in i:(n - 1)){
      r <- 0.5 * (abs(i)^H2 + abs(j)^H2 - abs(j - i)^H2)
      r <- r/n^H2
      matcov[i, j] <- r
      matcov[j, i] <- matcov[i, j]
    }
  }

  L <- chol(matcov)
  Z <- rnorm(n - 1)
  fBm <- t(L) %*% Z
  fBm <- c(0, fBm)

  drop(fBm)
}
```

Fución para calcular el filtro de Liu-West.

```
lw <- function(y, n) {
  # calcula Liu-West particle filter
  #
  # Entrada:
  # -------
  #   y: vector de observaciones
  #   n: numero de particulas
```

```
#
# Salida:
# ------
#   X: simulacion de la variable latente
#   Y: simulacion de la variable observada
#   theta: estimacion de los parametros

X <-  matrix(nrow = n, ncol = N)
X[, 1] <- rtmvnorm(n,
                   mean = sd(diff(y))/sqrt(1/N), # mean Hull pag 288
                   sigma = 1,
                   lower = 0,
                   upper = 1)

Yhist <-  matrix(nrow = n, ncol = N)
Y <- rtmvnorm(n,
              mean = mean(y),
              sigma = var(y),
              lower = 0,
              upper = Inf)
Yhist[,1] <- Y

thetaVector <- rtmvnorm(n,
                        mean = c(0, 0, 0.75, 0),
                        sigma = diag(c(0.0005, 0.001, 0.05, 0.0000015)),
                        lower = c(0, 0, 0, 0),
                        upper = c(1, 1, 1, 1))

a <- (3*0.99 - 1)/(2*0.99)

for (t in 2:N) {
  print(paste("t =", t))

  #1. Prior point estimates
  mi <- propagaXsinError(X[ , t-1], thetaVector[ , 1], 0)
  mPar <- apply(thetaVector, 2, mean)
  m <- a*thetaVector + (1 - a)*mPar

  # 2. Sample an auxiliar
  w <- discProb( propagaYsinError(Y, mi, m[ , 4]) , y[t])
  indices <- sample(1:n, size = n, replace = TRUE, prob = w)

```

```
50    X[, t-1] <- X[indices, t-1]
51    Y <- Y[indices]
52    Yhist[,t-1] <- Y
53    thetaVector <- thetaVector[indices, ]
54    w <- w[indices]
55
56    # 3. Sample a new parameter
57    mPar <- apply(thetaVector, 2, mean)
58    m <- a*thetaVector + (1 - a)*mPar
59
60    V <- cov(thetaVector)
61
62    diagV <- (1-a^2)*V
63    for(j in 1:n){
64      thetaVector[j, ] <- rtmvnorm(1,
65                                   mean = m[j, ],
66                                   sigma = diagV,
67                                   lower = c(0, 0, 0, 0),
68                                   upper = c(1, 1, 1, 1))
69    }
70
71    for(j in 1:n){ # ciclo particulas
72      # para el H se genera todo el camino, pero se toma la entrada
73      dBH <- diff(cholFBM(t+1, thetaVector[j, 3], 0))
74      X[j, t] <- propagaX(X[j, t-1], thetaVector[j, 1], 0, thetaVector[j, 2], dBH[t])
75    }
76
77    #Y_ant <- Y
78    Y <- propagaY(Y, X[ , t], thetaVector[ , 4])
79
80    # 5. Evaluate the corresponding weight
81    w2 <- discProb(Y, y[t]) / w
82    indices <- sample(1:n, size = n, replace = TRUE, prob = w2)
83
84    Y <- Y[indices]
85    Yhist[,t] <- Y
86    X[, t] <- X[indices, t]
87    thetaVector <- thetaVector[indices, ]
88
89    cat("Promedio:", apply(thetaVector, 2, mean, na.rm = T), "\n")
90    cat("Sd:", apply(thetaVector, 2, sd, na.rm = T), "\n")
91  }
```

```
92
93  return(list(X = X, Y = Yhist, theta = thetaVector, m = m, V = V))
94 }
```

Función para calcular el filtros de partículas boostrap.

```
1  filtroParticulas <- function(y, theta, n){
2    # Calcula Filtro de particulas
3    # Algoritmo 1, pag 12 Jacob
4    #
5    # Entradas:
6    #   y: vector de observaciones
7    #   theta: parametros
8    #   n: numero de particulas
9    #
10   # Salida:
11   #   verosimilitud
12   #   X: simulacion de la variable latente
13   #   w: pesos
14
15   X <-  matrix(nrow = n, ncol = N)
16   X[,1] <- rtmvnorm(n,
17                     mean = sd(diff(y))/sqrt(1/N),
18                     sigma = 1,
19                     lower = 0,
20                     upper = 1) # mean Hull pag 288
21
22   Yhist <-  matrix(nrow = n, ncol = N)
23   Y <- rtmvnorm(n,
24                 mean = mean(y),
25                 sigma = var(y),
26                 lower = 0,
27                 upper = Inf)
28
29   dBH   <- matrix(nrow = n, ncol = N)
30
31   for (i in 1:n) {
32     dBH[i, ] <- diff(cholFBM(N+1, theta$H, 0))
33   }
34
35   w <- dnorm(Y, mean = y[1], sd = 1)
36
37   verosimilitud <- log(calculaVerosimilitud(w))
```

```
38
39  for(t in 2:(N)){
40    indices <- sample(1:n, size = n, replace = TRUE, prob = w)
41
42    X[, t-1] <- X[indices, t-1]
43    Y <- Y[indices]
44    Yhist[,t-1] <- Y
45    w <- w[indices]
46
47    verosimilitud <- verosimilitud + log(calculaVerosimilitud(w))
48
49    X[, t] <- propagaX(X[, t-1], theta$alpha, 0, theta$beta, dBH[, t])
50
51    Y <- propagaY(Y, X[, t], theta$mu)
52    Yhist[, t] <- Y
53
54    w <- dnorm(Y, mean = y[t], sd= 1)
55  }
56
57  return(list(verosimilitud = verosimilitud, X = X, Y = Yhist))
58 }
```

Función para calcular el filtro de partículas marginal Metropolis-Hasting.

```
1  pmmh <- function(y, nP, M, theta0 = NA, sigmaMat = NULL, sigmaMat_p = NULL){
2    # Calcula Particle Marginal Metropolis-Hastings
3    # Algoritmo 3, pag 13 Jacob
4    #
5    # Entradas:
6    #   y: vector de observaciones
7    #   nP: numero de particulas
8    #   M: tamanno de la muestra de PMMH
9    #
10   # Salida:
11   #   thetaVector: muestra
12
13
14   if (is.null(sigmaMat)) {
15     sigmaMat = diag(c(0.0005, 0.0005, 0.05, 0.00005))
16   }
17
18   X_list <- vector("list", M)
19   Y_list <- vector("list", M)
```

```
thetaVector <- matrix(nrow = M, ncol = 4)
lb <- c(0, 0, 0, 0)
ub <- c(1, 1, 1, 1)

if (is.null(sigmaMat_p)) {
  sigmaMat_p <- sigmaMat
}

# 1:
if(anyNA(theta0)){
  meanIni <- c(0.05, 0.05, 0.51, 0.0020)
} else {
  meanIni <- theta0
}

thetaAnt <- rtmvnorm(1,
                     mean = meanIni,
                     sigma = sigmaMat,
                     lower = lb,
                     upper = ub)

thetaVector[1, ] <- thetaAnt
thetaAnt <- split(thetaAnt, c('alpha', 'beta', 'H', 'mu'))

# 2:
fp <- filtroParticulas(y, thetaAnt, nP)
verAnt <- fp$verosimilitud
XAnt <- fp$X
YAnt <- fp$Y

thetaAnt <- unlist(thetaAnt, use.names = F)

# 3:
for(t in 1:M){
  print(paste("t =", t))
  sink("logFile.txt", append = T)
  print(paste("t =", t))
  sink()

  # 4:
  thetaP <- rtmvnorm(1,
                     mean = thetaAnt,
```

```
                    sigma = sigmaMat_p,
                    lower = lb,
                    upper = ub)

  thetaP <- split(thetaP, c('alpha', 'beta', 'H', 'mu'))

  # 5:
  fp <- filtroParticulas(y, thetaP, nP)
  ver <- fp$verosimilitud
  X <- fp$X
  Y <- fp$Y

  thetaP <- unlist(thetaP, use.names = F)

  # 6:
  num <- (dtmvnorm(thetaP,
                   mean = meanIni,
                   sigma = sigmaMat,
                   lower = lb,
                   upper = ub,
                   log = T)
          + ver)

  den <- (dtmvnorm(thetaAnt,
                   mean = meanIni,
                   sigma = sigmaMat,
                   lower = lb,
                   upper = ub,
                   log = T)
          + verAnt)

  a <- num - den

  # 7:
  if(log(runif(1)) <= a){
    thetaVector[t, ] <- thetaP
    thetaAnt <- thetaP
    verAnt <- ver
    X_list[[t]] <- X
    XAnt <- X
    Y_list[[t]] <- Y
    YAnt <- Y
```

```
    } else {
      thetaVector[t, ] <- thetaAnt
      X_list[[t]] <- XAnt
      Y_list[[t]] <- YAnt
    }

    cat("Actual:  ", thetaVector[t, ], "\n")
    cat("Promedio:", apply(thetaVector, 2, mean, na.rm = T), "\n")
    cat("Sd:", apply(thetaVector, 2, sd, na.rm = T), "\n")

    sink("logFile.txt", append = T)
    cat("Actual:  ", thetaVector[t, ], "\n")
    cat("Promedio:", apply(thetaVector, 2, mean, na.rm = T), "\n")
    cat("Sd:", apply(thetaVector, 2, sd, na.rm = T), "\n")
    sink()
  }

  return(list(thetaVector = thetaVector, X_list = X_list, Y_list = Y_list))
}
```

Función para calcular el filtro de partículas secuencial aumentado MCMC.

```
SMCMC_filtroParticulas_parallel_v2_4p <- function(y, nn, burn=10000, seed=42){
  # Calcula Filtro de particulas
  # Algoritmo 1, pag 12 Jacob
  #
  # Entradas:
  #   y: vector de observaciones
  #   nn: numero de particulas
  #   burn: periodo de quema
  #
  # Salida:
  #   X: matriz con los caminos n x N
  #   Y: estimacion de Y en tiempo N
  #   thetaMCMC: vector que la muestra de las distribucion posterior


  # setup parallel backend to use many processors
  cores <- parallel::detectCores()

  # numero de observaciones
  N = length(y)

```

```
# numero de clustes
k <- 1

# se inicializa X, Y
X <-  matrix(nrow = nn, ncol = N+1)
X[,1] <- rtmvnorm(nn,
                  mean = sd(diff(y))/sqrt(1/N),
                  sigma = 1,
                  lower = 0,
                  upper = 1) # mean Hull pag 288

Yhist <-  matrix(nrow = nn, ncol = N)
Y <- rtmvnorm(nn,
              mean = mean(y),
              sigma = var(y),
              lower = 0,
              upper = Inf)

# se inicializa theta
meanTheta <- c(0.0, 0.0, 0.75, 0.0)
sigmaMat <- diag(c(0.0005, 0.001, 0.05, 0.0000015))
sigmaMat_p <- diag(c(0.06, 0.05, 0.05, 0.05))

thetaMCMC <- rtmvnorm(nn,
                      mean = meanTheta,
                      sigma = sigmaMat,
                      lower = c(0, 0, 0, 0),
                      upper = c(1, 1, 1, 1))

# se estima la mixtura gausiana
postTheta <- ajustaMMG(thetaMCMC, k)

# se fijan semillas para reproducir resultados
RNGkind("L'Ecuyer-CMRG")

# ciclo camino del proceso
for(tt in 1:N){
  print(paste0("t = ", tt))
  sink("logFile.txt", append = T)
  print(paste0("t = ", tt))
  sink()

```

```
64      # se fijan semillas para reproducir resultados # 42 seed
65      set.seed(seed * tt)
66
67      # en cada tiempo se hace n MCMC
68      thetaMCMC_l <- parallel::mclapply(1:nn, FUN = AMCMC_4p, tt = tt, #
69                                                              burn = burn,
70                                                              thetaMCMC = thetaMCMC,
71                                                              sigmaMat_p = sigmaMat_p,
72                                                              XX = X,
73                                                              Y = Y,
74                                                              y = y,
75                                                              postTheta = postTheta,
76                                        mc.cores = cores,
77                                        mc.set.seed = TRUE)
78
79
80      thetaMCMC <- do.call(rbind, thetaMCMC_l)
81
82      # se estima la mixtura gausiana
83      postTheta <- ajustaMMG(thetaMCMC, k)
84
85      # se calculan los caminos del proceso de memoria larga
86      dBH   <- matrix(nrow = nn, ncol = tt)
87      for (i in 1:nn) {
88        dBH[i, ] <- diff(cholFBM(tt+1, thetaMCMC[i, 3], 0))
89      }
90
91      X[, tt+1] <- propagaX(X[,tt], thetaMCMC[,1], 0, thetaMCMC[,2], dBH[,tt])
92      Y <- propagaY(Y, X[,tt+1], thetaMCMC[,4])
93
94      Yhist[, tt] <- Y
95
96      print(apply(thetaMCMC, 2, mean))
97      sink("logFile.txt", append = T)
98      print(apply(thetaMCMC, 2, mean))
99      sink()
100
101   }
102
103   return(list(X=X, Y=Y, Yh=Yhist, thetaMCMC=thetaMCMC))
104 }
```

Funciones auxiliares que se usan en el código de filtro de partículas secuencial aumentado MCMC.

```
probMixturaGausiana_4p <- function(y, mmg) {
  comp <- length(mmg$lambda)
  prob <- 0
  for (i in 1:comp) {
    resultado <- try({ mmg$lambda[i] * dtmvnorm(y,
                                                mean = mmg$mu[[i]],
                                                sigma = diag(diag(mmg$sigma[[i]])),
                                                lower = c(0, 0, 0, 0),
                                                upper = c(1, 1, 1, 1)) },
                     silent = T)
    if ("try-error" %in% class(resultado) ) {
      resultado <- mmg$lambda[i] * mvtnorm::dmvnorm(y, mean = mmg$mu[[i]], sigma = mmg$
          sigma[[i]])
    }
    prob <- prob + resultado
  }
  return(prob)
}


ajustaMMG <- function(datos, k){
  datosScale <- scale(datos)
  grupos <- kmeans(datosScale, k, nstart = 10)
  pesos <- as.vector(table(grupos$cluster) / sum(table(grupos$cluster)))
  df <- as.data.frame(datos) %>%
          mutate(g=grupos$cluster) %>%
          group_by(g) %>%
          nest() %>%
          arrange(g)
  promedios <- lapply(df$data, function(x) as.vector(apply(x, 2, mean)))
  matCov <- lapply(df$data, function(x) unname(cov(x)))
  return(list(lambda=pesos, mu=promedios, sigma=matCov))
}


AMCMC_4p <- function(iMCMC, tt, burn, thetaMCMC, sigmaMat_p, XX, Y, y, postTheta) {
    for (k in 1:burn) {

      # probabilidades de propuesta theta
```

```
39      qTheta <- tmvtnorm::rtmvnorm(1,
40                                   mean = thetaMCMC[iMCMC,],
41                                   sigma = sigmaMat_p,
42                                   lower = c(0, 0, 0, 0),
43                                   upper = c(1, 1, 1, 1))
44
45      # se calcula el camino del proceso de memoria larga
46      dBH <- diff(cholFBM(tt+1, qTheta[3], 0))
47      XX[iMCMC, tt+1] <- propagaX(XX[iMCMC, tt], qTheta[1], 0, qTheta[2], dBH[tt])
48      Yaux <- propagaY(Y[iMCMC], XX[iMCMC, tt+1], qTheta[4])
49      qw <- dnorm(Yaux, mean = y[tt+1], sd = 1)
50      qPostTheta <- probMixturaGausiana_4p(qTheta, postTheta)
51
52      # probabilidades de theta
53      dBH <- diff(cholFBM(tt+1, thetaMCMC[iMCMC, 3], 0))
54      XX[iMCMC, tt+1] <- propagaX(XX[iMCMC, tt], thetaMCMC[iMCMC, 1], 0, thetaMCMC[iMCMC,
            2], dBH[tt])
55      Yaux <- propagaY(Y[iMCMC], XX[iMCMC, tt+1], thetaMCMC[iMCMC, 4])
56      w <- dnorm(Yaux, mean = y[tt+1], sd = 1)
57      probPostTheta <- probMixturaGausiana_4p(thetaMCMC[iMCMC, ], postTheta)
58
59      # se obtiene sucesor
60      a <- max(0, min(1, (qw * qPostTheta) / (w * probPostTheta)), na.rm=T)
61      if (runif(1) <= a) {
62        thetaMCMC[iMCMC, ] <- qTheta
63      }
64    }
65
66  return(thetaMCMC[iMCMC, ])
67 }
```

El siguiente código se le agrega al filtro de partículas secuencial aumentado MCMC para realizar un remuestreo.

```
1   # se selecciona las particulas con mas peso con respecto a la observacion
2   w <- discProb(Y, y[tt+1])
3
4   w <- w /sum(w)
5   neff <-  1 / sum(w^2)
6   if(neff >= nn/10){
7     indices <-  sysResampling(w)
8
```

```
    auxTheta <- thetaMCMC[indices, ]
    if (nrow(unique(auxTheta, MARGIN = 1)) >= nparUm) {
      Y <- Y[indices]
      X[, tt+1] <- X[indices, tt+1]
      thetaMCMC <- auxTheta

      print(paste("resampling --", "neff:", neff, "-- n theta:", nrow(unique(thetaMCMC,
          MARGIN = 1))))
      sink("logFile.txt", append = T)
      print(paste("resampling --", "neff:", neff, "-- n theta:", nrow(unique(thetaMCMC,
          MARGIN = 1))))
      sink()
    }
}
```

REFERENCIAS BIBLIOGRÁFICAS

[1] Andrieu, C.; Doucet, A.; Holenstein, R. (2010) "Particle markov chain monte monte carlo methods". *Journal of the Royal Statistical Society*. Serie B Vol 72 No 3: 269–342.

[2] Beran, J. (1994) *Statistics for Long-Memory Processes*. Chapman and Hall, New York.

[3] Bracher, J.; Ray, E.; Gneiting, T.; Reich, N. (2020). "Evaluating epidemic forecasts in an interval format". Preprint, https://arxiv.org/pdf/2005.12881.pdf.

[4] Brockwell, P.; Davis, R. (2016) *Introduction to Time Series and Forecasting*, 3ra Edición. Springer, Switzerland.

[5] Cheridito, P.; Kawaguchi, H.; Maejima, M. (2003) "Fractional Ornstein-Uhlenbeck processes". *Electronic Journal of Probability* Vol 8 No 3: 1–14.

[6] Chronopoulou, A.; Viens, F. (2010) "Estimation and Pricing under long-memory stochastic volatility". *Annals of Finance* Vol 8 No 2-3: 379–403.

[7] Chronopoulou, A.; Viens, F. (2012) "Stochastic Volatility and Option Pricing with Long-Memory in Discrete and Continuous Time". *Quantitative Finance* Vol 12 No 4: 635–649.

[8] Coeurjolly, J. (2000) "Simulation and identification of the fractional brownian motion: a bibliographical and comparative study". *Journal of Statistical Software* Vol 5 No 7.

[9] Comte, F.; Renault, E. (1998) "Long Memory in Continuous-time Stochastic Volatility Models". *Mathematical Finance* Vol 8 No 4: 291–323.

[10] Deng, Z.; Zhang, X.; Tian, T.; (2016) "Copula particle filter algorithm for inferring parameters of regulatory network with noisy observation data". *IEEE International conference on bioinformatics and biomedicine* 570–575.

[11] Dahlin, J.; Schön, T. (2015) "Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models". Preprint, https://arxiv.org/pdf/1511.01707.pdf.

[12] Doucet, A; de Freitas, J.; Gordon, N. (2001) "An Introduction to Sequencial Monte Carlo Methods". En Doucet, A; de Freitas, J.; Gordon, N. (eds) *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.

[13] Fan, Y.; Huang, G.; Baetz, B.; Li, Y.; Huang, H. (2017) "Development of a copula-based particle filter (CopPF) approach for hydrologic data assimilation under consideration of parameter interdependence". *Water Resources Research* Vol 53 No 6:4850–4875.

[14] Gneiting, T.; Raftery, A. (2007) "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association* Vol 102 No 477:359–378.

[15] Hersbach, H. (2000) "Decomposition of the Continuous Ranked Probability Score Ensemble Prediction Systems". *Weather Forecasting* Vol 15 559–570.

[16] Hol, J. (2004) "Resampling in particle filters". Internship report Linköping University.

[17] Hoff, P. (2009) *A First Course in Bayesian Statistical Methods*. Springer- Verlag, New York.

[18] Hull, J. (2009) *Options, Futures, and other Derivatives*, 7ma Edición. Pearson Prentice Hall, New Jersey.

[19] Jacob, P; Alavi, S; Mahdi, A; Payne, S; Howey, D. (2018) "Bayesian Inference in Non-Markovian State-Space Models With Applications to Battery Fractional-Order Systems". *IEEE Transactions on Control Systems Technology* Vol 26 no. 2 pp. 497–506.

[20] Javvad, M; Dass, S; Asirvadam, V. (2019) "An augmented sequential MCMC procedure for particle based learning in dynamical systems". *Signal Processing* vol 160 pp. 32-44.

[21] Kaarakka, T. (2015) "Fractional Ornstein-Uhlenbeck Processes". *Tampere University of Technology* Vol 1338.

[22] Kitagawa, G; Sato, S. (2001) "Monte Carlo Smoothing and Self-Organising State-Space Model". En Doucet, A; de Freitas, J.; Gordon, N. (eds) *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.

[23] Lai, T; Bukkapatanam, V. (2013) "Adaptive Filtering, Nonlinear State-Space Models, and Applications in Finance". En Zeng, Y; Wu, S. (eds) *State-Space Models Applications in Economics and Finance*. Springer-Verlag, New York.

[24] Liu, J.; West, M. (2001) "Combined Parameter and State Estimation in Simulation-Based Filtering". En Doucet, A; de Freitas, J.; Gordon, N. (eds) *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.

[25] Mikosch, T. (1998) *Elementary Stochastic Calculus with finance in View*. World Scientific, Singapore.

[26] Pitt, M.; Shephard, N. (2001) "Auxiliary Variable Based Particle Filter". En Doucet, A; de Freitas, J.; Gordon, N. (eds) *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.

[27] R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

[28] Rios, M.; Freitas, H. (2013) "The Extended Liu and West Filter: Parameter Learning in Markov Switching Stochastic Volatility Models". En Zeng, Y.; Wu, S (eds) *State- Space Models Applications in Economics and Finance*. Springer-Verlag, New York.

[29] Robert, C. (2007) *The Bayesian Choice*, 2da Edición. Springer-Verlag, New York.

[30] Robert, C.; Casella, G. (2004) *Monte Carlo Statistical Methods*, 2da Edición. Springer-Verlag, New York.

[31] Rubinstein, R.; Kroese, D. (2008) *Simulation and the Monte Carlo Method*, 2da Edición. John Wiley & Sons, New Jersey.

[32] Särkkä, S. (2013) *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge.

[33] Tong, Z. (2012) *Option Pricing with Long Memory Stochastic Volatility Models*. Degree of Master of Science, University of Ottawa, Ottawa.

[34] Wikipedia contributors. (2020, November 17). S&P 500 Index. In Wikipedia, The Free Encyclopedia. Retrieved 23:19, November 28, 2020, from https://en.wikipedia.org/w/index.php?title=S%26P_500_Index&oldid=989247990