UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

Bayesian estimation in generalized autoregressive conditional heteroskedasticity models

Estimación bayesiana de modelos autorregresivos de heteroscedasticidad condicional generalizados

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Eva Romero Ramos

Directores

Juan Miguel Marín Diazaraque
María Teresa Rodríguez Bernal

Madrid, 2016

Bayesian estimation in generalized autoregressive conditional heteroskedasticity models. Estimación bayesiana de modelos autorregresivos de heteroscedasticidad condicional generalizados.

Universidad Complutense de Madrid

Eva Romero Ramos

Directores de tesis:
Dr. Juan Miguel Marín Diazaraque
Dra. María Teresa Rodríguez Bernal

November 9, 2015

Agradecimientos

Quiero dedicar unas líneas a expresar mi más profundo y sincero agradecimiento a todas las personas que con su ayuda y apoyo han colaborado en la realización de esta tesis. Y es que son muchas las personas que han estado a mi lado en estos años, ayudándome y dándome su apoyo y confianza. Como se suele decir, *si viajas solo llegarás antes, pero en compañía llegarás más lejos.* Yo siento que con este trabajo he recorrido un camino largo y es gracias a vosotros.

Sin duda debo empezar por la Dra. María Teresa Rodríguez Bernal y el Dr. Juan Miguel Marín Diazaraque, que asumieron la codirección de este trabajo y desde el principio se han volcado conmigo en todos los sentidos. Gracias de corazón por vuestros consejos y profesionalidad, pero sobre todo por vuestro cariño.

Debo dar las gracias también a mi hijo Alex por haberle dado un sentido muy especial a mi vida, dándome cada día la sonrisa que necesito para poner los pies en la tierra y entender lo verdaderamente importante de la vida.

A mi marido Miguel, mi compañero de sueños desde hace ya muchos años. Llevamos toda una vida luchando juntos y siempre siento que estamos empezando. Gracias por estar ahí siempre.

A mis padres por el cariño y esfuerzo que pusieron en mi educación y a mis hermanos porque siempre están ahí cuando hace falta.

Al Dr. Jose Ramos Rodrigo, mi primo, por creer en mí más de lo que yo misma he creído y mostrarme el mundo de la docencia y la investigación, que se ha convertido en mi forma de vida.

A la Dra. Eva Ropero Moriones mi compañera y amiga, porque sin sus consejos y apoyo sería imposible haber llegado hasta aquí.

A mi amiga la Dra. Noelia de Pablo Torres, con la que puedo contar siempre para lo que necesite. Gracias por tus consejos y tu cariño.

A mis compañeros de la Facultad de ciencias sociales de la Universidad Europea, porque a pesar de las dificultades del día a día somos una gran familia de la que me siento orgullosa de formar parte.

Gracias a todos, de corazón, porque sin vosotros, de un modo u otro, no sería quien soy.

Contents

List of Figures

Preface

This thesis was motivated by the interest in analysing and modelling financial time series which have specific characteristics, as the so called stylized facts. The time series models that are more suitable for analysing financial series are GARCH and COGARCH models. These models have been largely used in the last years to model assets, indexes or exchange rates, probably due to the special treatment that they apply to volatility.

It is well known that volatility is essential to understand the nature of financial series, and in the last years researchers have focused on modelling them. Subsequently, both models GARCH and COGARCH include two equations, one for the process (considering the returns in the case of GARCH models, and the observed values in the case of COGARCH ones) and other equation for the process variance or the volatility.

In this thesis I propose several algorithms, based on different Bayesian methods, which can be used to estimate the model parameters for both, GARCH and COGARCH models, obtaining valuable solutions in simulation studies and over real time series.

In Chapter 1, I introduce the concepts and models used in this work, including the definition of ARCH, GARCH and COGARCH models an their properties. In this chapter, the behaviour of COGARCH models when parameters change is analysed in order to understand how parameters rule the behaviour of the series, and how to estimate them properly.

In Chapter 2 several methodologies to estimate COGARCH parameters have been reviewed, highlighting the approximation to the COGARCH(1,1) model of

Maller et al. (2008). This approximation is specially relevant in this work, because it is the base to compute the Bayesian estimators for the COGARCH model parameters proposed in this thesis. The first approach, based on a MCMC methodology is introduced, addressing two different situations: when we have several sample paths of the COGARCH process, and when using only one trajectory. I analyse the good behaviour of the estimates by using simulated and real data sets. Codes have been developed in R and they are included in an Appendix.

Chapter 3 introduces the estimators for GARCH and COGARCH model parameters based on a data cloning methodology. This method constitutes a combination of Classical and Bayesian frameworks which provides estimates valid for both, Classical and Bayesian researchers. It uses a simple Monte Carlo algorithm to approximate maximum likelihood estimates. I use this technique in GARCH and COGARCH model parameters, and I analyse their accuracy by using simulated and real data. The obtained results are published in Marín et al. (2015).

A Hamiltonian Monte Carlo approach is introduced in Chapter 4 for GARCH and COGARCH model parameters. The goal of this methodology is to provide a Bayesian algorithm which obtains, in a more efficient way, the posterior distribution of the parameters of interest. The estimates obtained by using this method may be considered equivalent to the obtained by standard MCMC algorithms, but computational times are significantly smaller, especially when we estimate parameters in the complex COGARCH models. I deal with the estimation of GARCH and COGARCH model parameters, with simulated and real data sets in order to asses their quality. Codes have been developed in R and they are included in an Appendix.

Finally, in Chapter 5, an Approximated Bayesian Computation (ABC) frame-

work has been applied in the same context. ABC background provides a good solution to estimate parameters in complex models, when the maximum likelihood function is unknown or difficult to approximate, like COGARCH models. In order to estimate GARCH model parameters I have developed two algorithms: an ABC and an ABC-MCMC scheme. But the most valuable technique introduced in this chapter is the ABC method for COGARCH model parameters, which provides efficient estimates without using the likelihood function. The R codes for all algorithms can be found in the Appendix.

0.1 Resumen

Esta tesis doctoral nace con el propósito de entender, analizar y sobre todo modelizar el comportamiento estadístico de las series financieras. En este sentido, se puede afirmar que los modelos que mejor recogen las especiales características de estas series son los modelos de heterocedasticidad condicionada en tiempo discreto, si los intervalos de tiempo en los que se recogen los datos lo permiten, y en tiempo continuo si tenemos datos diarios o datos intradía.

Con esta finalidad, en esta tesis se proponen distintos estimadores bayesianos para la estimación de los parámetros de los modelos GARCH en tiempo discreto (Bollerslev (1986)) y COGARCH en tiempo continuo (Klüppelberg et al. (2004)).

En el capítulo 1 se introducen las características de las series financieras y se presentan los modelos ARCH, GARCH y COGARCH, así como sus principales propiedades.

Mandelbrot (1963) destacó que las series financieras no presentan estacionariedad y que sus incrementos no presentan autocorrelación, aunque sus cuadrados sí están correlacionados. Señaló también que la volatilidad que presentan no es constante y que aparecen *clusters* de volatilidad. Observó la falta de normalidad de las series financieras, debida principalmente a su comportamiento leptocúrtico, y también destacó los efectos estacionales que presentan las series, analizando cómo se ven afectadas por la época del año o el día de la semana. Posteriormente Black (1976) completó la lista de características especiales incluyendo los denominados *leverage effects* relacionados con cómo las fluctuaciones positivas y negativas de los precios de los activos afectan a la volatilidad de las series de forma distinta.

Engle (1982) definió el modelo ARCH como medio para incluir algunas de

las características presentes en las series financieras, centrándose sobre todo en la modelización de la volatilidad. Para ello este modelo incluye en su definición dos expresiones, una que modeliza el proceso conductor del modelo y otra, su proceso varianza. Pero a pesar de incluir muchas de las particularidades de las series financieras como la heterocedasticidad, la falta de autocorrelación de los incrementos o las distribuciones leptocúrticas, existen otras características como los *leverage effects*, que no aparecen en los procesos ARCH. Por otro lado, estos imponen importantes restricciones sobre los parámetros, de modo que se hace necesario incluir un gran número de términos para modelizar la volatilidad adecuadamente.

Para solventar estos inconvenientes se establecieron los modelos GARCH (Bollerslev (1986)), que añaden respecto a los modelos ARCH un término autorregresivo en su expresión de la varianza. Estos modelos mejoran la modelización de la volatilidad sin necesidad de incluir un gran número de términos, pero no recogen los *leverage effects* y no son capaces de reproducir las colas altas de las distribuciones de las series financieras reales. No obstante, en la actualidad son los más utilizados en modelización financiera porque a pesar de estos inconvenientes resultan de gran utilidad en términos de predicción de series financieras. En esta tesis doctoral, entre otros temas, trataremos el enfoque bayesiano de estos modelos.

Por su parte, los modelos COGARCH amplían los modelos de heterocedasticidad condicionada en tiempo discreto, permitiendo modelizar situaciones en las que los procesos no presentan intervalos de tiempo equidistantes, sino que se producen en intervalos de tiempo de diferente longitud, como ocurre en las series de datos diarios o intradía. De entre los modelos GARCH en tiempo continuo que se han establecido, se puede destacar el COGARCH de Klüppelberg et al. (2004) por

presentar buenas características para la modelización de series financieras, dado que tanto el proceso conductor como su proceso varianza comparten la misma fuente de aleatoriedad en términos de un proceso de Lévy. En esta tesis doctoral se estudia también el enfoque bayesiano de estos modelos.

En el capítulo 2, tras presentar una revisión de los métodos de estimación más relevantes que se han propuesto para los modelos COGARCH, se introduce un modelo bayesiano basado en una metodología MCMC y en la parametrización de Maller et al. (2008), donde se estiman sus parámetros a partir de una o varias trayectorias del proceso.

Primero se presenta la estimación basada en el método de los momentos, desarrollada por Haug et al. (2005). Este método ofrece buenos resultados en la práctica, pero presenta dos inconvenientes importantes. El primero es que, para poderlo utilizar, debemos asumir que los datos se presentan en tiempos equidistantes, y el segundo es que requiere el uso de varias trayectorias del proceso para obtener estimaciones de calidad, hecho que en la práctica es poco realista.

Posteriormente se presenta el modelo de Maller et al. (2008) donde se discretiza el modelo COGARCH en intervalos de tiempo no equidistantes, de modo que para intervalos de tiempo equidistantes resulta equivalente a un proceso GARCH. Si se toma el límite sobre el número de intervalos no equidistantes de una sucesión de particiones sobre el tiempo (llevando así el proceso a tiempo continuo) el modelo converge al COGARCH. En esta tesis doctoral se utiliza este enfoque como base para los métodos de estimación bayesianos. A partir de esta discretización Maller et al. (2008) determinan una función de pseudo-verosimilitud, y se estiman los parámetros del COGARCH mediante su maximización. Sin embargo, para que las estimaciones sean más precisas, al igual que

en el estimador basado en el método de los momentos, es conveniente también el uso de múltiples trayectorias.

Finalmente, se presenta el método de estimación bayesiana de Müller (2010), que éste compara con el método de pseudo-máxima verosimilitud de Maller et al. (2008). El algoritmo que plantea genera los tiempos de salto y los parámetros del modelo, incluyendo el valor inicial de la volatilidad y la intensidad del salto en cada iteración. El resultado es un algoritmo bastante complejo, en el que es conveniente partir de valores iniciales que no estén muy alejados de los verdaderos valores a estimar. En este sentido, el autor propone como valores iniciales para los parámetros, las estimaciones de pseudo-máxima verosimilitud de Maller et al. (2008).

La aportación original en este capítulo se centra en un modelo bayesiano basado en un algoritmo estándar MCMC, para la estimación de los parámetros del modelo COGARCH. Al estar basado en la misma aproximación al COGARCH usada por Maller et al. (2008), este algoritmo a diferencia del de Müller (2010), no estima los intervalos de tiempo en los que aparece el proceso, dado que en observaciones reales podemos suponer que son valores fijos conocidos previamente. Para simplificar, se asume, al igual que en la aproximación de Maller et al. (2008), que la intensidad de los saltos y el valor inicial de la volatilidad son constantes. Se obtienen buenos resultados en la práctica que sólo requieren el uso de una trayectoria del proceso. Se estudian, no obstante, los resultados obtenidos con varias trayectorias del proceso y se comprueba empíricamente que los resultados son similares a los obtenidos partiendo de una única trayectoria.

En el capítulo 3 se describe la metodología *data cloning* y se aplica en el caso de los modelos GARCH y COGARCH.

En el procedimiento de estimación utilizando el método *data cloning*, se aproxima el estimador de máxima verosimilitud mediante el uso de algoritmos MCMC en modelos jerárquicos complejos donde los métodos estándar de máxima verosimilitud no funcionan adecuadamente.

La idea principal en estos procedimientos es clonar los datos (la serie financiera en el contexto que nos ocupa) y asumir que son distintas trayectorias simuladas del mismo modelo, que han dado lugar al mismo resultado. En esta situación, la función de verosimilitud resulta ser la que tendríamos para una trayectoria, pero elevada al número de trayectorias clonadas (número de clones). Para estimar los parámetros del modelo se aplica un algoritmo MCMC sobre las trayectorias clonadas, obteniéndose así las distribuciones a posteriori de los parámetros a estimar.

En el modelo GARCH se presenta un algoritmo en cinco pasos sencillos que, utilizando un pequeño número de clones, es capaz de estimar los parámetros del modelo de un modo eficiente, como se comprueba en un estudio de simulación. Los resultados obtenidos pueden compararse con los obtenidos aplicando el algoritmo Metropolis-Hasting planteado por Nakatsuma (1998). Se puede comprobar que ambos presentan una eficiencia similar, aunque en el caso de utilizar la metodología *data cloning* los estimadores se deben interpretar en un contexto frecuentista, que permite calcular intervalos de confianza estándar.

El algoritmo basado en el *data cloning* se aplica también en una serie de datos reales: los rendimientos semanales del *Nasdaq* 100 desde enero del año 2000 hasta noviembre de 2012. Tras comprobar que el modelo que mejor se ajusta al comportamiento de la serie es un GARCH(1,1), se estiman sus parámetros y se comparan las estimaciones con las obtenidas mediante el método MCMC de

Nakatsuma (1998). Se obtienen valores similares tanto de las estimaciones de los parámetros como de sus desviaciones estándar.

Por otro lado, se aplica la metodología *data cloning* en el modelo COGARCH. En este sentido, teniendo en cuenta la complejidad del modelo, el algoritmo resultante es más eficiente que el de Maller et al. (2008), puesto que permite obtener estimaciones de máxima verosimilitud y sus correspondientes intervalos de confianza, partiendo únicamente de una trayectoria del proceso.

Se comprueba empíricamente que las estimaciones obtenidas y sus desviaciones típicas, son similares a las que mostraba el estimador basado en un método MCMC presentado en el capítulo 2. En este caso, se obtienen intervalos de confianza que incluyen en todos los casos a los verdaderos valores de los parámetros. Se aplica también el algoritmo de *data cloning* a la estimación de los parámetros del COGARCH(1,1) para modelizar la serie de datos diarios del índice *Nasdaq* 100, desde enero de 2008 hasta diciembre de 2012. De nuevo, los resultados obtenidos son similares a los que muestra el estimador basado el métodos MCMC presentado en el capítulo 2.

En el capítulo 4 se describe la metodología *Hamiltonian Monte Carlo* (HMC) y se usa como base para la implementación de dos nuevos métodos de estimación de los parámetros, en los casos del GARCH y del COGARCH.

Los algoritmos y resultados obtenidos mediante el uso de la metodología data cloning para la estimación de los parámetros de los modelos GARCH y COGARCH están publicados en Marín et al. (2015).

Cuando se trata de aplicar algoritmos MCMC sobre modelos complejos, como son los modelos GARCH y COGARCH, la convergencia de las cadenas puede resultar muy lenta. La metodología HMC resulta ser una mejora sustancial que

se basa en la definición de un vector de variables auxiliares denominadas *variables momento*.

La metodología HMC se puede resumir en 3 pasos: un primer paso en el que se generan los valores iniciales de los parámetros del modelo y de las variables momento; un segundo paso en el que se implementa el denominado algoritmo *Leapfrog*, que depende de la derivada del logaritmo de la función de densidad a posteriori y de un factor de escala; y un tercer paso Metropolis-Hasting para la aceptación o rechazo de los valores obtenidos en el paso anterior.

Los algoritmos implementados en los modelos GARCH y COGARCH son similares y se obtienen en ambos casos buenos resultados tanto con datos simulados como con datos reales. Para comparar los resultados obtenidos con los correspondientes a los capítulos 2 y 3, se utilizan los mismos datos. Resulta de especial interés el algoritmo presentado para la estimación de los parámetros del COGARCH, en el que la metodología HMC resulta más eficiente y permite reducir los tiempos de estimación.

Finalmente, en el capítulo 5 se presenta una metodología relativamente reciente, denominada *Approximate Bayesian Computation* (ABC), y se utiliza para la estimación de los parámetros de los modelos GARCH y COGARCH.

La idea básica de los métodos ABC consiste en generar los parámetros a partir de unas distribuciones a priori, con ellos simular una muestra de datos y tomar los parámetros como válidos si la muestra obtenida resulta suficientemente parecida a los datos originales, basándose en unos estadísticos de bondad de ajuste.

Este algoritmo ABC se puede complementar incluyendo procedimientos de aceptación o rechazo mediante pasos Metropolis-Hasting en términos de un algoritmo denominado MCMC-ABC. En ambos casos, la definición de los estadísticos

que miden la aproximación entre las muestras simuladas y la muestra original resulta ser el componente del algoritmo más relevante.

En este capítulo se presentan dos algoritmos para la estimación de los parámetros del modelo GARCH: un método ABC y otro MCMC-ABC. En modelos GARCH, como estadísticos de ajuste se proponen la media, la varianza y la desviación mediana absoluta. Sin embargo, es conveniente acotar los valores propuestos de las distribuciones a priori para reducir los tiempos de computación. Con este fin, se acotan las distribuciones a priori en un entorno de los estimadores de pseudo-máxima verosimilitud.

La metodología ABC resulta de especial interés en la estimación de modelos COGARCH, dado que no es necesario utilizar las funciones de pseudo-verosimilitud para obtener los estimadores. En este caso se propone un algoritmo ABC para la estimación de los parámetros del modelo, basado en la comparación de cinco estadísticos: la mínima volatilidad del proceso, el rango de la volatilidad del proceso, la correlación entre los cuadrados de los valores consecutivos del proceso, la varianza y la desviación mediana absoluta.

De nuevo resulta conveniente acotar los valores propuestos de las distribuciones a priori basándose, en este caso, en un entorno de los estimadores obtenidos por el método de los momentos. Cuando se aplica el método a datos simulados, se obtienen errores estándar menores que los obtenidos en el resto de metodologías presentadas en los capítulos previos.

Con datos reales se puede comprobar que las estimaciones obtenidas están en línea con las del resto de metodologías presentadas en los capítulos anteriores, aunque de nuevo se puede apreciar cómo los errores estándar son sustancialmente inferiores en la mayoría de los parámetros.

Se concluye, por tanto, que las diferentes metodologías bayesianas presentadas en esta tesis doctoral ofrecen soluciones válidas para la estimación de los parámetros en modelos de heterocedasticidad condicionada en tiempo discreto, como los modelos GARCH y en tiempo continuo como los modelos COGARCH.

0.2 Abstract

This thesis has the purpose of understand, analyse and specially modelize the statistical behaviour of financial series. In this sense, it can be affirmed that the models which include the financial series stylized facts better are the conditional heteroskedascity models in discrete or continuous times, depending on the series nature.

With this objective, this thesis proposes different bayesian estimators to estimate the model parameters of discrete time GARCH models (from Bollerslev (1986)) and continuous time COGARCH models (from Klüppelberg et al. (2004)).

In Chapter 1 the special characteristics of financial series are described and ARCH, GARCH and COGARCH models with their properties are presented.

Mandelbrot (1963) highlighted that financial series do not present stationarity and their increments do not present autocorrelation, but their squares are correlated. He also pointed out that the volatility they present is not constant and appear in *volatility clusters*. Mandelbrot (1963) identified the absence of normality of financial series, principally due to their leptokurtic shape. He highlighted the seasonal effects this series present, analyzing how their are affected by the time of the year or the day of the week. Afterward Black (1976) completed the list of stylized facts including the called *leverage effects* related to the different

way the positive and negative fluctuations of assets prices affect to the volatility of the series.

Engle (1982) defined the ARCH model to include some the characteristics of financial series, especially focus on modelizing the volatility. With this purpose this model includes in its definition two expressions, one to modelize the driving process and one more to the variance process. But despite of including several stylized facts as heteroskedasticity, lack of autocorrelation in the increments or the leptokurtic distributions, some others like the leverage effects are not incorporated in ARCH processes. On the other hand, these processes impose important restrictions over parameters that make necessary including a big number of terms to modelize volatility properly.

To solve these issues GARCH models where established by Bollerslev (1986) adding to ARCH models an autorregresive term in variance expression. These models improve the volatility modelization without requiring a large number of terms, but do not include leverage effects and they are not able to reproduce the heavy tails real series present. Anyway, nowadays they are the most largely used to modelize financial series because, despite of their disadvantages they provide good enough predictions. In this thesis, the bayesian approach to this model will be discussed among other issues.

COGARCH models extend the conditional heteroskedasticity models in discrete time, to let modelizing processes with non equally spaced times as daily or intra-day series. Among all continuous times GARCH models can be highlighted the COGARCH model of Klüppelberg et al. (2004) because it presents good characteristics to modelize financial series, considering that the driving process and variance share the same random source in terms of a Lévy process. In this thesis

is also analyzed the bayesian approach to these models.

In Chapter 2, after reviewing the most relevant estimation methods proposed for COGARCH models, a bayesian estimator based on a MCMC methodology according to Maller et al. (2008) parametrization is proposed, where model parameters are estimated from one or several process sample paths.

First the estimation based on power moment method from Haug et al. (2005) is described. This method provides good results in practice but present two important shortcomings. The first is that to use it, it is necessary assume that data has equally spaced times and the second is that it requires using several sample paths of the process to obtain high quality estimations and this is not realistic in practice.

Afterward, Maller et al. (2008) model is reviewed. In this approach the COGARCH model is discretized in unequally spaced times and if equally spaced times are considered, the resulting process is equivalent to a GARCH process. Considering the limit over the number of unequally intervals of a succession of partitions over the times (making the process continuous), the model converges to COGARCH. In this thesis, this approach is used as the base of the introduced bayesian estimation methods. From this discretization Maller et al. (2008) obtained a pseudo likelihood function and estimate the COGARCH parameters with its maximization. But to obtain accurate estimations with this method, as with the power moments estimator, is advisable to use several sample paths of the process.

Finally the estimation method from Müller (2010) is analysed and compared with the pseudo-likelihood method of Maller et al. (2008). This algorithm proposes generate the jump times, the model parameters, the initial volatility value

and the jump size in each iteration. The result is a complex enough algorithm in which is advisable start from initial values for the parameters not far from the real values to be estimated. In this sense, the author propose as initial values the pseudo-likelihood estimations from Maller et al. (2008).

The original contribution in this chapter is a bayesian model based on a standard MCMC algorithm to estimate the COGARCH model parameters. As it is based on the approach of Maller et al. (2008), this algorithm does not estimate the time intervals of the process, as Müller (2010) algorithm does, considering that, in real series the time points could be assumed as previously known. To simplify, as in Maller et al. (2008) approximation it is also assume that the jumps size and the initial value of volatility are constant. This algorithm provides good estimations in practice by using only one sample path of the process. Anyway, the obtained results by using several sample paths from the same process are also analysed and it is empirically proved that the results are similar to the obtained by using just one trajectory.

In Chapter 3 data *cloning methodology* is described and applied to GARCH and COGARCH models.

In the estimation process using *data cloning*, the maximum likelihood estimator is approximated by using MCMC algorithms in complex hierarchical models where standard maximum likelihood does not work properly.

The main idea of these procedures is cloning data (financial series in our context) and assume that we have several sample paths simulated from the same process which have the same result. In this situation, the likelihood function is the likelihood function which will be obtained with one trajectory but to the power of the number of cloned sample paths (number of clones). To estimate the

model parameters, MCMC algorithm is applied over the cloned trajectories to obtain the posterior distributions of the parameters to be estimated.

In the GARCH model a five steps algorithm is proposed which, using an small number of clones is able to estimate the model parameters in an efficient way, as it is proved in an empirical simulation study. The obtained results can be compared with the obtained applying the Metropolis-Hasting algorithm from Nakatsuma (1998). It can be proved that both estimators present a similar efficiency, even though data cloning estimators should be interpreted in a frequentist context which let calculating standard confidence intervals.

The algorithm base on data cloning method is also applied over a real financial series: the weekly returns of *Nasdaq* 100 from January of 2000 to November of 2012. After verifiying that the model that better fit to the behaviour of this series is a GARCH(1,1), its parameters are estimated and the estimations are compared with the obtained by using with MCMC method from Nakatsuma (1998). The obtained parameters values and its standard deviations are quite similar by using both methods.

On the other hand, *data cloning* method is applied to COGARCH model. In this sense, considering the model complexity, the proposed algorithm result to be more efficient than the obtained by Maller et al. (2008), because it lets obtaining maximum likelihood estimations and its corresponding confidence intervals from just one sample path of the process.

It is proved empirically that the obtained estimations and its standard deviations are similar to the showed by the estimator based on the MCMC method proposed in chapter 2. In this case, confidence intervals are obtained and include in all cases the true parameter values. This data cloning algorithm is also applied

to estimate the model parameters of the COGARCH(1,1) model, which modelize the daily returns of *Nasdaq* 100 index from January 2008 to December 2012. Again the obtained results are similar to the estimations by using the MCMC algorithm introduced in chapter 2.

The algorithms and the obtained results by data cloning method to estimate model parameters of GARCH and COGARCH models are published in **?**.

In chapter 4 *Hamiltonian Monte Carlo* (HMC) is described and used as base to develop two new estimation methods for GARCH and COGARCH parameters.

When we apply MCMC algorithms over complex models as GARCH and COGARCH models, chains convergence use to be slow. HMC methodology prove to be a substantial improve based on the definition of a vector of auxiliary variables called *momentum variables*.

HMC methodology can be summarized in 3 steps: A first step to generate the initial values for model parameters and for *momentum variables*; a second step to implement the called *Leapfrog* algorithm, which depends on the derivative function of the logarithm of the posterior density function and on a scale factor; and a third step to include the accept-reject Metropolis-Hasting method for the previously obtained values.

The GARCH and COGARCH algorithms are similar and provide good results over simulated and real series. To compare the results with the obtained in chapters 2 and 3, the same series are used. It is specially interesting the proposed algorithm to estimate COGARCH parameters because HMC method provides a more efficient algorithm which reduces the estimation time.

Finally, in chapter 5 a rather new methodology called *Approximate Bayesian*

Computation (ABC) is described and used to estimate the model parameters of GARCH and COGARCH models.

The main idea of ABC methods is generate the parameter values from the prior distributions, simulate with them a sample path and consider the proposed values as good if the obtained trajectory is similar enough to the original data, considering several goodness of fit statistics.

This basic ABC algorithm can be completed by including accept-reject procedures by means of Metropolis-Hasting steps in term of an algorithm called MCMC-ABC. In both cases, the statistics definition to measure the nearness between the simulated and original sample paths is the most relevant part of the algorithm.

In this chapter, two algorithms to estimate GARCH parameters are proposed: one based on an ABC method and another based on a MCMC-ABC method. In GARCH models, as goodness of fit statistics, mean, variance and median absolute deviation are proposed. Nevertheless, it is advisable to bound the proposed values from prior distributions to reduce the computation time. In order to do it, prior distributions are bounded by using the pseudo-maximum likelihood estimations.

ABC method is specially interesting to estimate COGARCH models, considering that it does not required using the pseudo-likelihood function to estimate. In this case, an ABC algorithm is proposed to estimate de model parameters based on comparing five statistics: the minimum process volatility, the range of the process volatility, the correlation between the squared consecutive process values, the process variance and the process median absolute deviation.

In this case is also advisable to bound the prior proposed values, based now on a range around the power moment estimations. Applying this method over simu-

lated data, smaller standard errors than with the rest of methodologies described in this thesis are obtained.

With real series can be verified that the obtained estimations are similar to the rest of proposed methods, in spite of, again it could be appreciated that standard errors are significantly smaller for most of parameters.

With everything it can be concluded that the different bayesian methods proposed in this thesis offer valid solutions to estimate model parameters in conditional heteroskedaticity models in discrete times like GARCH and in continuous times like COGARCH.

Chapter 1

Introduction to COGARCH Models

1.1 Introduction

The origin of stock market is not clearly established, and it could be said that it is as old as the commercial sector, which from its origins addressed transactions like nowadays in the actual stock markets. It may be considered that stock market was born as an institution in the middle-ages festivals of west Europe, at the end of the fifteenth century, although earlier, in the 12th century in France, there were people who managed and standardized debts from farming communities in representation of banks. They could be called the first *brokers*.

Despite the interest of this kind of transactions, finance was not considered as a science until the nineteenth century, when companies began to grow faster and they merged due to the industrial revolution. In those days, stock market started working as it does nowadays and researchers started analysing the behaviour of the markets and their financial series.

From the 50th decade of the twentieth century, financial markets were largely analysed in economy. At the beginning, researchers based their theories in the *efficient market hypothesis* (EMH). This hypothesis basically maintains that new information is at last known by all investors, almost at the same time, and consequently is immediately incorporated to the market prices. This means that an efficient market includes prices, whose values incorporate all the relevant information and are updated immediately and independently. When modelling financial series, EMH means that prices returns are independent and identically distributed random variables (iid) which implies that market has no memory, and past returns are not useful to predict the future ones. This paradigm was largely used in financial analysis due to the simplicity of the iid returns and the linear relation among variables. But it was obvious that the forecast capability of

models based on this hypothesis was no very good. Thus, in the 70th decade, the most commonly used technique to analyse financial series was the Box-Jenkins methodology, which was established by Box et al. (1970). In this methodology, time series were analysed by ARIMA models, which are easy to implement due to their simplicity, but they show the restriction of assuming marginal and conditional constant variances.

Computation advances at the end of the twentieth century, allowed researchers in economy and finance to analyse the financial market behaviour under new perspectives. The new paradigm was based in a non-linear structure. This new way of thinking started in the middle 80th decade, when researches from physics questioned the EMH paradigm. In those days stock markets showed a strong volatility which made necessary to look for new theories to explain the movements of financial markets, considering the behavior of market agents. The new framework was based in non-linear models as fractal models, chaos theory or GARCH models, and it was quite a revolution in the quantitative finance field.

Fractal mathematics includes models with long term persistence which is a generalization of short term memory ARIMA models. These models became specially important with the introduction of the fractional differentiation operator by Hosking (1981).

Chaos theory can be understood as the randomness generated by deterministic systems, by means of the sensibility of chaotic systems to their initial conditions. Chaotic systems may be considered as a mix of randomness and determinism which provide a new point of view to analyse events, that may be recognized initially as random but they evolve with time. Chaotic systems are based in simple rules that, as times goes by, provide complex behaviour, indistinguishable

with respect to the randomness. Therefore, sometimes it is used the term *pseudo-randomness* to talk about this kind of behaviour.

By the other hand, non-linear stochastic modelling addresses about the non-stationary of increments assuming the presence of heteroskedasticity in the model. This approach includes ARCH, GARCH and COGARCH models and their derivatives. The special importance that volatility has nowadays when analysing financial series, make these models being specially important, because they let analyse both, returns and volatilities, at the same time. In this PhD thesis, different estimators for GARCH and COGARCH models parameters are analysed using different Bayesian methodologies.

1.2 Financial series and their stylized facts

Financial series present some particular characteristics which make them quite difficult to analyse. Mandelbrot (1963) observed and analysed these characteristics, which he called *stylized facts* and, in the following years, other authors continued to analyse the difficulties that financial series present, in terms of prediction tasks.

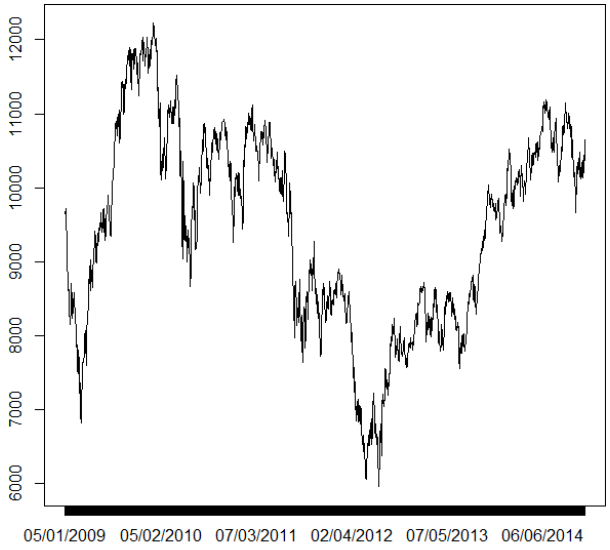

Figure 1.1: Ibex-35 evolution between 2009 and 2014

As a typical example to illustrate the stylized facts of financial series, in figure 1.1 it is shown the evolution of the Ibex-35 index between 2009 and 2014. It may be observed that its behaviour could be described as *wild*, and for this reason it is expected to be very difficult to undertake predictions. The volatility of this series is represented in figure 1.2, and it is very clear that it changes in time, making it necessary to include volatility in the model.

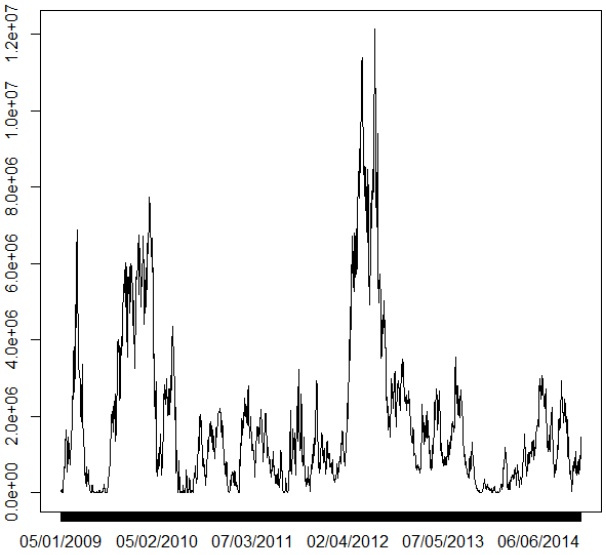

Figure 1.2: Volatility of Ibex-35 between 2009 and 2014

Despite the fact that most financial series have a common performance depending on the nature of the asset, they may be measured in terms of different units. For example, stock prices from companies are measured in the monetary units of the market that they are quoted, but indexes, or interest rates are usually measured in points. Difference among units could be a problem when we try to compare among time series. Furthermore, it is generally more relevant to analyse the profitability of the assets instead of the values they have. In order to do it, instead of considering the price, it is better to define the increment of the price, in terms of

$$x_n = \log\left(\frac{p_n}{p_{n-1}}\right)$$

where p_n is the price of the assets in the market in time n, and p_{n-1} is the price in the previous time. Using this transformation, the evolution of the Ibex-35 between 2009 and 2014 is shown in figure 1.3.

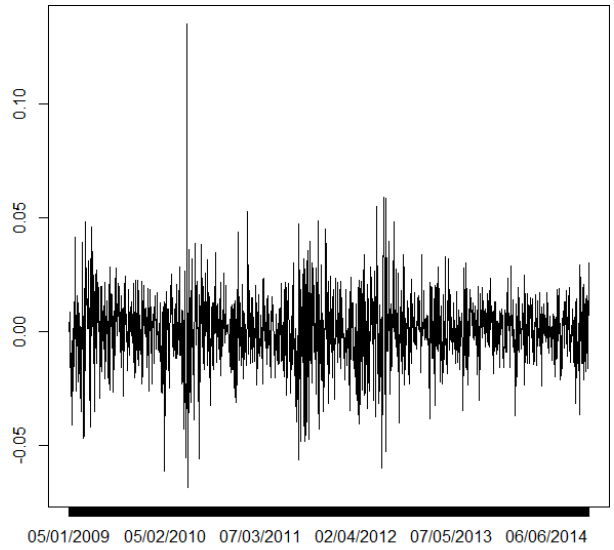

Figure 1.3: Returns of Ibex-35 between 2009 and 2014

These increments show clearly the wild behaviour of random walks, where values could be positive or negative and the volatility changes along the time. This fact illustrates the random walk characteristically behaviour of financial series.

Figure 1.4 shows the volatility of returns, which is slightly different than the volatility of stock prices, because it does not accumulate the previous jumps and it just shows the daily volatility.

The stylized facts are statistical irregularities that were put forward by Mandelbrot (1963), but in the following years other authors found out new characteristics of financial series to be included as stylized facts. They are quite difficult to reproduce with standard stochastic models, but stylized facts appear in a large list of financial series of different nature like interest rates, exchange rates, stocks and option prices.

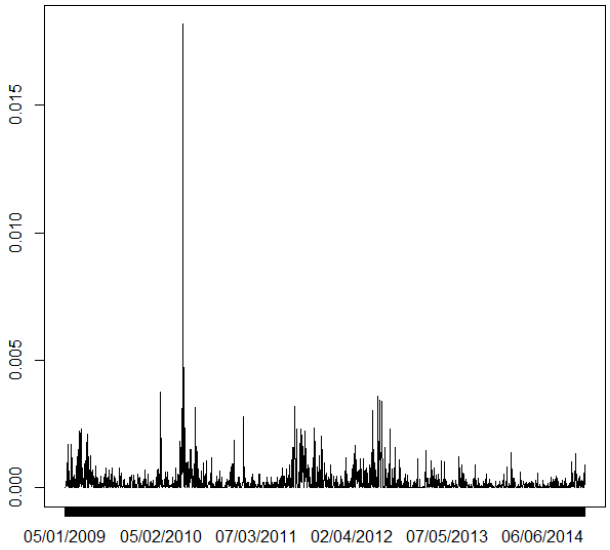

Figure 1.4: Volatility of returns of Ibex-35 between 2009 and 2014

Some of the most important characteristics of financial series or stylized facts are listed below:

- **Non-stationarity**

  Sample paths of financial series have usually an extremely random behaviour (see i.e. figures 1.1 or 1.3) which is quite similar to a random walk. Nevertheless, in most cases the second order moments of this kind of series are stationary. This is an useful property which may be used to analyse and model them.

- **Non autocorrelated price variations**

  Financial series usually present small autocorrelations, which make their correlogram looks quite similar to that obtained in a white noise series. This type of behaviour is observed when undertaking daily, weekly or monthly

series, but in case of smaller time periods (minutes or even seconds) autocorrelation becomes higher. This is due to the called *microstructure effects*.

- **Squared price returns autocorrelated**

  In financial series, while returns are uncorrelated, their square values and their absolute values present significant autocorrelation which decay slowly. Absolute returns present higher autocorrelation than square returns.

- **Volatility clustering**

  Returns usually present volatility clustering, that is, large changes tend to be followed by large changes and small changes tend to be followed by small changes. For example, in Ibex-35 between 2009 and 2014 (Figure 1.4, and Figure 1.3), there are periods of high volatility and periods of low volatility. These two different situations do not present any periodic behaviour. Volatility clustering shows that financial series are heterocedastic and it is not possible to apply the standard Box-Jenkins methodology (Box et al. (1970)).

- **Heavy-tails distributions**

  The empirical distribution plot of daily returns shows a non-Gaussian behaviour. In general, when applying to daily returns any normality test, the normality hypothesis is usually rejected. The main reason arises because financial series are usually leptokurtic and they present fat tails.

  Figure 1.5 shows the histogram of the standardized daily returns of Ibex-35 between 2009 and 2014, and the curve of the standard normal distribution. It may be observed that the distribution of data has higher kurtosis than the normal distribution.

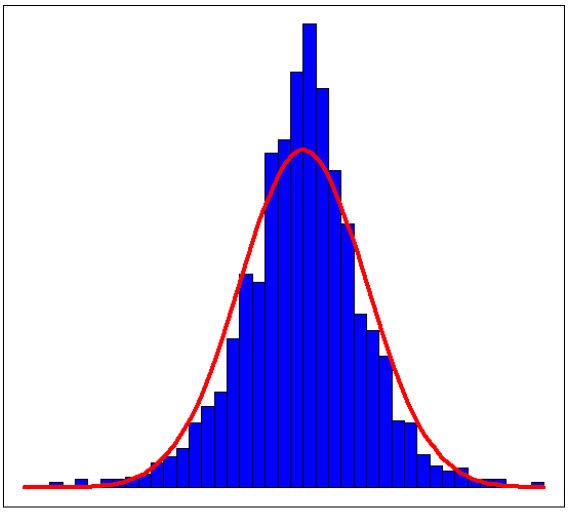

Figure 1.5: Density function of Ibex between 2009 and 2014

As data frequency increases, for example in monthly series, the distribution is closer to the normal distribution.

- **Leverage effects**

  Leverage effects concepts were introduced by Black (1976). They play a role in the way which positive and negative past values affect to volatility.

  Black (1976) observed the relation between the decreasing of assets prices and high volatilities. Usually, negative returns increase volatility more than positive ones, namely, volatility has higher correlation with negative returns.

- **Calendar effects**

  It is well known that the time at which returns are observed, as the day of a week or the season in a year, affects them significantly. For example, at the end of a week or the previous days before holidays, the stock prices use to go slightly down. This is mainly due to risk aversion of investors who know

that, if the market is going to be closed for a large period of time, unexpected events can change dramatically the behaviour of the stock market.

Seasonality can influence prices, because volatility uses to increase after a long period of time when markets are closed. This is because information cumulated during breaks is usually reflected in prices. By the other hand, events that happen when markets are open have more influence in prices than those in periods of time when markets are closed.

In this way, in order to assume the previous characteristics, stylized facts must be included in models to undertake the analysis of financial series.

1.3 ARCH and GARCH Models

In this section, we introduce the ARCH and GARCH models that can include time-varying volatility and address to analyse time series which present heteroskedasticity. They also include most of the stylized facts presented in the previous section. Furthermore, they show a simple linear structure that supplies a clear basis to the analysis and estimation of financial time series.

1.3.1 ARCH Models

ARCH models were introduced by Engle (1982) who proposed to model the volatility as a linear function of the past returns. In this way, ARCH models enable to consider uncorrelated time series assuming positive correlation of square returns. It is assumed that the volatility, σ_n^2 is a random variable and returns depend on a sequence of random errors, independent and identically distributed ϵ_n. The definitions of ARCH(1) and ARCH(q) processes are

Definition 1. ***ARCH(1) process.***

Given that the volatilities σ_n^2 are stationary and independent random variables and, assuming that ϵ_n, called innovations, is a sequence of random variables, independently and identically distributed, the process Y_n is an ARCH(1) model if

$$Y_n = \sigma_n \epsilon_n \quad \textit{where} \quad \sigma_n^2 = \beta + \lambda Y_{n-1}^2, \quad n \in \mathbb{N}$$

where β and λ are parameters satisfying $\beta > 0$, $\lambda \geq 0$ to guarantee a positive marginal variance. □

The model presents conditional autoregressive heteroskedastidity of first order.

Definition 2. ***ARCH(q) process.***

Given that the volatilities σ_n^2 are stationary and independent random variables and, assuming that ϵ_n is a sequence of random variables, independently and identically distributed, the process Y_n is an ARCH(q) model if

$$Y_n = \sigma_n \epsilon_n \quad \textit{where} \quad \sigma_n^2 = \beta + \sum_{i=1}^{q} \lambda_i Y_{n-1}^2, \quad n \in \mathbb{N}$$

where β and λ_i are parameters satisfying $\beta > 0$, $\lambda_i \geq 0$ to guarantee a positive variance. Now it is also required that $\sum_{i=1}^{q} \lambda_i < 1$ to ensure that the process σ_n^2 is stationary. □

With this structure, it may be noticed that a high value of past squared returns, Y_{n-i}^2, provides a high conditional variance in time n, σ_n^2, which involves obtaining high values of Y_n^2 with a higher probability, because $Y_n^2 = \sigma_n^2 \epsilon_n^2$.

Therefore, in ARCH processes large returns are probably followed by big ones. This implies that the squared returns present correlation and high variance too.

This characteristic can explain the volatility clustering that real financial series show in practice.

The process ϵ_n is defined as a standardized white noise, where $E[\epsilon_n] = 0$ and $E[\epsilon_n^2] = 1$.

Properties of ARCH models

- In ARCH models, the conditional and marginal means of Y_n are equal to zero,

$$\begin{aligned} E[Y_n] &= E[\sigma_n \epsilon_n] = E[\sigma_n]E[\epsilon_n] = 0, \\ E[Y_n|F_{n-1}] &= E[\sigma_n|F_{n-1}]E[\epsilon_n] = 0, \end{aligned}$$

  where F_{n-1} represents all the information related to the series in $n-1$ and assuming that volatilities and innovations are independent.

- The conditional variance is

$$E[Y_n^2|F_{n-1}] = \sigma_n^2 = \beta + \sum_{i=1}^{q} \lambda_i Y_{n-1}^2.$$

- The marginal variance is

$$Var[Y_n] = E[Y_n^2] = E[E[Y_n^2|F_{n-1}]] = \beta + \sum_{i=1}^{q} \lambda_i E[Y_{n-i}^2]$$

  If the series Y_n is stationary, the variance can be obtained as

$$Var[Y_n] = \frac{\beta}{1 - \sum_{i=1}^{q} \lambda_i}$$

  In this way, the condition for the process to be stationary is $\sum_{i=1}^{q} \lambda_i < 1$.

- The distribution of Y_n is leptokurtic.

- Y_n does not present autocorrelation, but it can be proved that an ARCH(q) process presents autoregressive dependence of order q among the squared returns given that

$$Y_n^2 = \beta + \lambda_1 Y_{n-1}^2 + \cdots + \lambda_q Y_{n-q}^2 + \eta_n$$

where η_n is a stationary white noise process with mean zero and constant variance.

Despite ARCH models include many of the stylized facts of financial series, they show some limitations from a practical point of view (see Tsay (2005)). Some of the shortcomings of ARCH models can be resumed by the following ideas:

- ARCH models assume that positive returns have the same effect over volatility than negative returns, as far as volatility depends on the squared past returns. But as Black (1976) remarked, in real financial series it may be observed that volatility is more affected by negative returns than by positive ones.

- ARCH models are quite restrictive in terms of its parameters. If we consider i.e. an ARCH(1) model, parameter λ_1^2 must be included in the interval $\left(1, \frac{1}{3}\right)$ in order to be finite the fourth order moment of the series. In case of higher order ARCH models, restrictions became more complex and estimation more difficult.

- To describe a volatility process properly, ARCH models require to include a high number of lags.

- ARCH models allow to describe the variance behaviour, but they do not explain the causes of this behaviour.

1.3.2 GARCH Models

In order to amend some of the shortcomings of ARCH models when modelling financial series, Bollerslev (1986) introduced GARCH models as an extension of them. This extension includes an autoregressive term to define the process variance.

Regarding GARCH models, the volatility of the series (σ_n^2) is assumed to be a random variable, and ϵ_n is a sequence of random variables, independent and identically distributed, with mean equal to 0 and variance equal to 1. The innovations ϵ_n are usually assumed to follow a normal or a t-Student distribution.

Definition 3. ***GARCH(1,1) process.***

Assuming that volatilities σ_n^2 are random variables and ϵ_n is a sequence of random variables, independently and identically distributed, then Y_n follow a GARCH(1,1) process model if

$$Y_n = \sigma_n \epsilon_n$$

where the volatility process is

$$\sigma_n^2 = \beta + \lambda Y_{n-1}^2 + \delta \sigma_{n-1}^2 \tag{1.1}$$

for $n \in \mathbb{N}$, $\beta > 0$, $\lambda \geq 0$, $\delta \geq 0$ and $\lambda + \delta \leq 1$. We consider also that ϵ_n are normally distributed $N(0,1)$. $\square$

If $\delta = 0$ the term σ_{n-1}^2 vanishes and it is obtained an ARCH(1) model.

In GARCH models, the volatility is also stochastic because of the dependence of σ_n^2 on Y_{n-1}. If $\lambda = \delta = 0$, Y_n is just a sequence of random variables with constant variance equal to β. To exclude this case it is necessary to set $\lambda + \delta > 0$.

As in ARCH models, a large value in Y_{n-1}^2 or σ_{n-1}^2 provides a higher variance σ_n. In this way, large squared returns use to be followed by also large squared returns, which produce the usual volatility clusters.

One of the advantages of GARCH over ARCH models it that GARCH models do not require a large number of lags to describe the series properly. In fact, a GARCH(1,1) model is equivalent to an ARCH(∞) model; the proof is obtained by using a recursive argument on the definition of the conditional variance

$$\sigma_n^2 = \beta + \lambda Y_{t-1}^2 + \delta\left(\beta + \lambda Y_{t-2}^2 + \delta\sigma_{n-2}^2\right) = \cdots = \sum_{i=0}^{\infty} \delta^i \beta + \lambda \sum_{i=1}^{\infty} \delta^{i-1} Y_{n-i}^2$$

The general GARCH models, that is, GARCH(p, q) models, are defined as follows

Definition 4. ***Strong GARCH(p, q) process.***

Assuming that volatilities σ_n^2 are random variables and ϵ_n is a sequence of random variables, independently and identically distributed, then Y_n follow a general strong GARCH(p, q) model if

$$Y_n = \sigma_n \epsilon_n$$

where the volatility process is

$$\sigma_n^2 = \beta + \sum_{i=1}^{q} \lambda_i Y_{n-i}^2 + \sum_{j=1}^{p} \delta_j \sigma_{n-j}^2 \tag{1.2}$$

for $n \in \mathbb{N}$, $\beta > 0$, $\lambda_i \geq 0$ ($i = 1, \ldots, q$), $\delta_j \geq 0$ ($j = 1, \ldots, p$), and $\sum_{i=1}^{\max(p,q)} (\lambda_i + \delta_i) < 1$. We consider also that ϵ_n are normally distributed $N(0, 1)$.$\square$

The last condition is necessary to guarantee the marginal variance to be positive and the existence of higher order moments. It implies that the marginal variance of Y_n is finite and the conditional variance is modified with time.

If $\delta_j = 0 \ \forall \ (j = 1, \ldots, p)$, terms σ_{n-j}^2 are null and it is obtained an ARCH(q) process.

Drost and Nijman (1993) introduced the semi-strong GARCH process just substituting Y_n^2 by $\sigma_{n-i}^2\epsilon_{n-i}^2$ in (1.2). Therefore, the variance of the semi-strong GARCH process is then,

$$\sigma_n^2 = \beta + \sum_{i=1}^{q} \lambda_i \sigma_{n-i}^2 \epsilon_{n-i}^2 + \sum_{j=1}^{p} \delta_j \sigma_{n-j}^2, \tag{1.3}$$

which it can also be written as

$$\sigma_n^2 = \beta + \sum_{i=1}^{r} \lambda_i(\epsilon_{n-i})\sigma_{n-i}^2, \tag{1.4}$$

where $n \in \mathbb{N}$, $r = \max(p,q)$, $\lambda_i = 0$ if $i > q$ $(i = 1, \ldots, q)$, $\delta_j = 0$ if $j > p$ $(j = 1, \ldots, p)$ and $\lambda_i(x) = \lambda_i x^2 + \delta_i$ for $i = 1, \ldots, r$.

From expression (1.4), it can be proved that the volatility in a GARCH process is the solution of an autoregressive equation with random coefficients. By the other hand, if $\beta_j = 0$ $(j = 1, \ldots, p)$ it is obtained an ARCH(q) process.

The main characteristics of GARCH models are included in the simplest model, GARCH(1,1), which is used in modelling most of financial series in practice.

Some of the main properties of the GARCH(1,1) model are shown in the next section.

Properties of the GARCH(1,1) model

- The conditional and marginal means of Y_n are equal to zero, as in ARCH models.

- The conditional variance is

$$E[Y_n^2|F_{n-1}] = \sigma_n^2 = \beta + \lambda Y_{n-1}^2 + \delta\sigma_{n-1}^2$$

- The marginal variance can be calculated as

$$\begin{aligned} Var(Y_n) &= E(Y_n^2) \\ &= E[E(y_n^2|F_{n-1})] \\ &= E[E(\sigma_n^2|F_{n-1})E(\epsilon_n^2|F_{n-1})] \\ &= E[\beta + \lambda Y_{n-1}^2 + \delta\sigma_{n-1}^2] \\ &= \beta + \lambda E[Y_{n-1}^2] + \delta E[\sigma_{n-1}^2] \\ &= \beta + \lambda Var[Y_{n-1}] + \delta E[\sigma_{n-1}^2] \end{aligned}$$

  As Y_n is a stationary process with constant variance σ^2, $Var(Y_n) = Var(Y_{n-1}) = \sigma^2$. Therefore $E(\sigma_n^2) = E(\sigma_{n-1}^2) = \sigma^2$.

  Substituting this term in the previous equation we obtain

$$\sigma^2 = \beta + \lambda\sigma^2 + \delta\sigma^2 \Leftrightarrow \sigma^2 = \frac{\beta}{1-\lambda-\delta}$$

  which requires that $\lambda + \delta < 1$, in order to obtain a stationary process.

- If $\epsilon_n \sim N(0,1)$, the kurtosis coefficient of Y_n is

$$kur(Y_n) = \frac{3[1-(\lambda+\delta)^2]}{1-(\lambda+\delta)^2-2\lambda^2)} > 3$$

  which requires that $1-(\lambda+\delta)^2-2\lambda^2) > 0$ so that the moment of order 4 is positive. Then, the distribution of Y_n will be leptokurtic, as the distribution of real financial series are.

- In a GARCH(1,1) process all moments of odd order of Y_n are zero, hence the distribution is symmetric. Furthermore, the tails of the distribution are heavier than the normal one, characteristic which is observed in real financial series.

- A GARCH(1,1) model presents a similar dependence for squared observations as ARCH(1,1) models,

$$Y_n^2 = \beta + (\lambda + \delta)Y_{n-1}^2 + \epsilon_n - \delta\epsilon_{n-1}$$

where ϵ_n is a sequence of random variables, independently and identically distributed with mean 0 and constant marginal variance $\epsilon_n = Y_n^2 - \sigma_n^2$.

By the other hand, $\lambda+\delta$ is called *persistence* in financial series and it usually has a value close to 1.

The correlation function, $\rho(k)$, is

$$\rho(k) = (\lambda + \delta)(k-1)\rho(1)$$

where

$$\rho(1) = \frac{\lambda(1 - \lambda\delta - \delta^2)}{1 - 2\lambda\delta - \delta^2}$$

When k increases, the autocorrelation of squared series decreases in a rate which depends of the persistence. If $\lambda + \delta$ is near to 1 it corresponds to a high persistence and the coefficients decrease slower.

Similarly as ARCH models, GARCH models also have some shortcomings. They can be resumed by the following ideas:

- In GARCH models, positive and negative returns present the same influence in the volatility, but in real financial series positive and negative returns are not symmetrical.

- Recent empirical studies show that GARCH processes do not achieve as heavy tails as the distribution of real financial series, even when using t-Student distributions for ϵ_n.

1.4 COGARCH Models

From the former Black and Scholes papers (Black and Scholes (1973)) it has been widely assumed that financial series may be modelled using continuous time models. As GARCH models present some of the most important stylized fact of the financial series, a continuous time version of them could improve the results obtained by standard GARCH models.

In discrete time models it is necessary to use a source of randomness included as a set of random variables, usually independent and identically normally distributed. In continuous time the role of an equivalent source of randomness is fulfilled by Brownian motions or Lévy processes.

A Brownian motion is a stochastic process originally introduced as an empirical way to analyse the erratic movement of coloured pollen grains in a liquid (see Revuz and Yor (1999)).

Definition 5. *A Brownian motion is a stochastic process* $\{B_t, t \geq 0\}$ *which satisfies the following conditions:*

(i) It starts at zero $B_0 = 0$ *almost surely.*

(ii) It has independent increments, which means that $B_{t+s} - B_t$ *is independent of* $\sigma(B_u : 0 \leq u \leq t)$ *for all* $0 \leq s, t < \infty$.

(iii) It has Gaussian increments, that is, $B_{t+s} - B_t \sim N(0, s)$ $\forall$ $0 \leq t, s < \infty$

(iv) It has continuous trajectories.

Properties of a Brownian motion

(*i*) The random variables $B_t - B_s$ and B_{t-s} are distributed as a normal distribution $N(0, t-s)$ if $s < t$. As the variance is the length of the interval, in Brownian motions larger intervals show larger fluctuations.

(*ii*) A Brownian motion is a Gaussian process. This means that its finite-dimensional distributions are multivariate Gaussian distributions.

(*iii*) As the process has independent increments

$$B_{t_1} - B_{t_0}, B_{t_2} - B_{t_1}, \ldots, B_{t_n} - B_{t_{n-1}}$$

are independent for any collection $0 \leq t_0 < \ldots < t_n < \infty$ and any $n \geq 0$.

(*iv*) The covariances are $cov(B_t, B_s) = min(s, t)$ for all $0 \leq t < s < \infty$

Lévy processes

Lévy processes take their name form the French mathematician Paul Lévy, and they are continuous time processes with independent and stationary increments. A Lévy processes can be defined as follows (see Tankov (2003)):

Definition 6. *Lévy processes*

A Lévy process is a stochastic process $(L_t)_{t \geq 0}$ that satisfies the following properties:

(i) It starts at zero $L_0 = 0$ almost surely.

(ii) It has independent and stationary increments.

(iii) It is stochastically continuous. This means that $\forall \epsilon > 0, lim_{s \to 0} P(|L_{t+s} - L_t| \geq \epsilon) = 0$. This condition does not imply that the sample path are continuous.

A characteristic of the Brownian motion is that it has continuous sample paths with probability one. This assumption can be relaxed by allowing jumps, but it is required the process to be a càdlàg process, that is:

- the process is right-continuous, $\lim_{s \downarrow t} L_s = L_t$
- the process has limits from the left, $L_{t^-} = \lim_{s \uparrow t} L_s$

with probability one.

The process of the jumps of the Lévy process $\Delta L_t = L_t - L_{t^-}$ is also a càdlàg process. It is expected these types of jumps to appear in financial economic time series.

Definition 7. *A random variable X is said to be infinitely divisible if there exists $n \in \mathbb{N}$ and a sequence of independent random variables $X_1^n, \ldots, X_n^n$ such that X has the same distribution as $X_1^n + \ldots + X_n^n$.*

An infinitely divisible random variable can be represented through its characteristic function as

$$\exp\{\psi(u)\} = E\left(e^{iuX}\right),$$

where $\psi(u)$ is the characteristic exponent of the process, for $u \in \mathbb{R}$ and $t > 0$. The process can be then characterized by its characteristic exponent which satisfies the Lévy-Khintchine formula.

The Lévy process L_t is infinitely divisible for every $t \geq 0$, taking

$$\begin{aligned} X_1^n &= L_{t/n} - L_0 \\ X_2^n &= L_{2t/n} - L_{t/n} \\ &\cdots \\ X_n^n &= L_t - L_{(n-1)t/n} \end{aligned}$$

The Lévy process L_t can be characterized in terms of the characteristic exponent of L_1. If $\phi_t(u) = E\left(e^{iuL_t}\right)$, it can be written in terms of the characteristic function of L_1, $\phi(u) = E\left(e^{iuL_1}\right)$ in the following way

$$\phi_t(u) = (\phi(u))^t .$$

Lévy-Khintchine representation of Lévy processes

The distribution of a Lévy process can be characterized by its characteristic function $\phi_L(u)$ given by

$$\phi_L(u) = E\left(e^{iuL(t)}\right) = \exp\{t\psi(u)\}$$

where $\psi(u)$ is the characteristic exponent of L_1 given by

$$\psi(u) = iau - \frac{\sigma^2 u^2}{2} + \int_{\mathbb{R}\setminus\{0\}} \left(e^{iux} - 1 - iux1_{\{|x|\leq 1\}}\right)\nu(dx)$$

where $a \in \mathbb{R}$, $\sigma \geq 0$ and ν is the Lévy measure of L_t, satisfying that $\nu(\{0\}) = 0$ and $\int min\{x^2, 1\}\nu(dx) < \infty$.

The triplet (a, σ^2, ν), or the characteristic exponent $\psi(u)$ of L_1, are usually used to characterized the process.

Lévy-Ito representation

Some examples of Lévy processes are:

1. The standard Brownian motion $\{B_t\}$.
2. The Brownian motion with drift $\{\mu t + \sigma B_t\}$.
3. The Poisson process $\{N_t\}$ with rate λ.

4. The compound Poisson process $\{\sum_{i=1}^{N_t} X_i\}$, where $\{N_t\}$ is a Poisson process with rate λ and X_i are independent and identically distributed random variables independent of each other.

Any linear combination of a Lévy process is again a Lévy process. In particular, the sum of a Brownian motion with drift and a compound Poisson process is a Lévy process. Except for the pure linear drift case $L_t = \mu t$, the Brownian motion is the only Lévy process with continuous sample paths. All other Lévy process have jumps.

If L_t is a Lévy process with triplet (a, σ^2, ν) and characteristic exponent $\psi(u)$ of L_1, then the Lévy-Ito representation of the process is

$$L_t = \mu t + \sigma B_t + L_t^d,$$

where $\mu t + \sigma B_t$ is a scaled Brownian motion with drift and L_t^d is an independent pure jump process. In particular, L_t can be decomposed into

$$L_t = L_t^1 + L_t^2 + L_t^3,$$

where

- $L_t^1 = \mu t + \sigma B_t$ is a scaled Brownian motion with drift with characteristic exponent of the form $iau - \frac{\sigma^2 u^2}{2}$.
- L_t^2 is a compound process on $\mathbb{R}\backslash(-1,1)$ with characteristic exponent

$$\int_{\mathbb{R}\backslash(-1,1)} \left(e^{iux} - 1\right) \nu(dx)$$

- L_t^3 is a square integrable martingale with an almost surely countable number of small jumps of magnitude less than unity, in a finite time interval, with characteristic exponent $\int_{(-1,1)\backslash\{0\}} (e^{iux} - 1 - iux)\, \nu(dx)$

If $\Delta L_t = L_t - L_{t^-}$ are the jumps of the Lévy process, then the counting process $N_t^B = card\{s \leq t : \Delta L_t \in B\}$ is a Poisson process for any Borel set in $\mathbb{R} \setminus \{0\}$ with rate $\nu(B)$. Then, $\nu(dx)$ is the intensity of jumps of size x, and the Lévy-Ito representation reduces to

$$L_t = \mu t + \sigma B_t + \sum_{s \leq t} \Delta L_t$$

with $\mu = a - \int_{|x| \leq 1} x\nu(dx)$, for $\nu(\mathbb{R}) < \infty$ where the standard Brownian motion is independent of the jumps of the process. The parameter μ is called the drift of the Lévy process.

The Lévy process has finite variation if and only if $\sigma = 0$, that is, there is no Brownian part. In this case

$$L_t = \mu t + \sum_{s \leq t} \Delta L_t.$$

1.4.1 COGARCH models

One of the first authors who dealt with a continuous version of a GARCH model was Nelson (1990) who defined a continuous GARCH model and derived a limiting GARCH(1,1) model by adding GARCH innovations up. The resulting model was driven by two different and independent Brownian motions, one as the driving process and the other one, present in the variance process. However, Wang (2002) proved that the diffusion limit of Nelson was not asymptotically equivalent to a GARCH process, considering the Le Cam's deficiency distance.

By the other hand Corradi (2000) modified the Nelson approximation to make it asymptotically equivalent to a GARCH process, in such a way in the resulting continuous GARCH model, the driving process and the variance process were driven by the same Brownian motion. But, a drawback of this model is that the

limiting probability process is not a random process which is a limitation to define stock prices properly. Duan (1997) obtained a diffusion limit for a more general GARCH process and Drost and Nijman (1993) proved that common GARCH models are not closed under temporal aggregation in the same way as semi-strong ARMA models. But they proved that weak GARCH models are closed under temporal aggregation.

Based on this research, Drost and Werker (1996) defined a continuous time process based on the class of weak GARCH models; some of the parameters were equivalent to other from a weak GARCH process, in such a way that estimation methods can be similar in both cases. Moreover, fourth moments are finite for the class of continuous-time weak GARCH nested processes, in the same way as in other models driven by two independent Lévy processes. Meddahi and Renault (2004) improved this model introducing the square-root stochastic autoregressive volatility models for both continuous and discrete times. Its main advantages are

(*i*) It allows asymmetries.

(*ii*) It is not necessary the fourth moment to be finite.

(*iii*) Discrete time square-root stochastic autoregressive volatility models are closed under temporal aggregation.

(*iv*) The discretization of continuous time square-root stochastic autoregressive volatility models, provides discrete time square-root stochastic autoregressive volatility models.

Kallsen and Taqqu (1998) introduced another approach for continuous-time GARCH models in which the driving process and the variance process were both driven by the same Brownian motion. When discrete times are taken for this

process, it is obtained a GARCH model. Kallsen and Taqqu (1998) also proved that the model is arbitrage free and complete, but the volatility is constant inside all time intervals $[t, t+1]$ although the model could be applied to determine pricing formulas for European options.

Kazmerchuk et al. (2005) introduced a continuous time GARCH process driven also by one Brownian motion, and they defined the process as a combination of volatility and price processes using a stochastic delay differential equation. Later, Lorenz (2006) obtained the solutions for the corresponding differential equation.

As far as this work is concerned, we will focus on the COGARCH model introduced by Klüppelberg et al. (2004). The advantage of this model is that it is driven by one single Lévy process which is responsible for the jumps of the process. As Bollerslev (2008) noticed, this model includes jumps and it exhibits a source of risk non-diversifiable, which is a remarkable characteristic of financial series.

The COGARCH (continuous-time GARCH) model is based on a background driving Lévy process and preserves the essential features of discrete-time GARCH processes. Recall that the volatility process of a GARCH(1,1) model is

$$\sigma_n^2 = \beta + \lambda Y_{n-1}^2 + \delta \sigma_{n-1}^2$$

where Y_n is of the form

$$Y_n = \sigma_n \epsilon_n$$

for $n \in \mathbb{N}$.

If the volatility is reorganize and iterate it is obtained

$$\sigma_n^2 = \beta \sum_{i=0}^{n-1} \prod_{j=i+1}^{n-1} \left(\delta + \lambda \epsilon_j^2\right) + \sigma_0^2 \prod_{j=0}^{n-1} \left(\delta + \lambda \epsilon_j^2\right)$$

Replacing the sum by an integral

$$\sigma_n^2 = \left(\beta \int_0^n \exp\left(-\sum_{j=0}^{\lfloor s \rfloor} \log\{\delta + \lambda \epsilon_j^2\} \right) ds + \sigma_0^2 \right) \exp\left(\sum_{j=0}^{n-1} \log\{\delta + \lambda \epsilon_j^2\} \right)$$

where $\lfloor s \rfloor$ is the largest integer not exceeding s.

Replacing ϵ_j by jumps of the Lévy process L_t and taking as parameters β, $\eta = -\log \delta$ and $\varphi = \frac{\lambda}{\delta}$, for a finite random variable σ_0^2 the volatility process is defined by

$$\sigma_t^2 = \left(\beta \int_0^t e^{X_s} ds + \sigma_0^2 \right) e^{-X_t}$$

for $t \geq 0$, where X_t is the auxiliary process

$$X_t = t\eta - \sum_{0<s\leq t} \log\{1 + \varphi(\Delta L_s)^2\} \tag{1.5}$$

for $t \geq 0$.

Then, the COGARCH(1,1) can be defined as the càdlàg process that satisfies the following stochastic differential equation

$$dG_t = \sigma_t dL_t$$

for $t \geq 0$ and $G_0 = 0$.

The process σ_t^2 satisfies the following stochastic differential equation

$$d\sigma_t^2 = (\beta - \eta \sigma_{t^-}^2) dt + \varphi \sigma_{t^-}^2 d[L, L]_t$$

where $[L, L]_t$ is the quadratic variation of the Lévy process

$$[L, L]_t = \sigma^2 t + \sum_{0<s\leq t} (\Delta L_s)^2 = \sigma^2 t + [L, L]_t^d$$

where $[L, L]_t^d = \sum_{0<s\leq t} (\Delta L_s)^2$ is the discrete part of the quadratic variation of L_t.

Regarding all the previous assumptions, we can define the COGARCH(1,1) process as follows.

Definition 8. *COGARCH(1,1) process*

Let $L = (L(t))_{t\geq 0}$ be a Lévy process with triplet (a, σ^2, ν). Given a finite random variable σ_0 independent of L_t, the COGARCH process $G = (G_t)_{t\geq 0}$ and the variance process $\sigma^2 = (\sigma_t^2)_{t\geq 0}$ are defined by the stochastic differential equations

$$dG_t = \sigma_t dL_t \tag{1.6}$$

$$d\sigma_t^2 = (\beta - \eta\sigma_{t^-}^2)dt + \varphi\sigma_{t^-}^2 d[L, L]_t \tag{1.7}$$

where $t > 0$, $G_0 = 0$, $\beta > 0$, $\eta > 0$, $\varphi \geq 0$ and $[L, L]_t$ is the quadratic variation of the Lévy process. □

1.4.2 Properties of COGARCH model

As Müller (2010) remarks, the COGARCH model captures many of the stylized facts that financial series present, as heavy tails, volatility jumps and volatility clusters. By the other hand, it incorporates an unique source of uncertainty, a Lévy process to drive both processes. The Lévy process can model also jumps of the process, which are often observed in real financial series.

We highlight the following properties of COGARCH models:

(*i*) The process G jumps at the same times as L does, and the size of its jumps is $\Delta G_t = \sigma_t \Delta L_t$ for $t \geq 0$. In this way, ΔL_t has the role of innovations in case of GARCH models.

(*ii*) The auxiliary process X_t is a spectrally negative Lévy process, with drift η, no Gaussian part and with Lévy measure

$$\nu_X([0,\infty)) = 0 \tag{1.8}$$

$$\nu_X((-\infty,-x]) = \nu(|y| \geq \sqrt{(e^x-1)\varphi}) \tag{1.9}$$

for $x > 0$.

(*iii*) If $\int_{\mathbb{R}} \log(1+\varphi x^2)\nu(dx) < \eta$ (if and only if $E(X_1) > 0$), then $\sigma_t^2 \xrightarrow[n\to\infty]{d} \sigma_\infty^2$, where σ_∞^2 is a finite random variable that satisfies

$$\sigma_\infty^2 \stackrel{d}{=} \beta \int_0^\infty e^{-X_t} dt.$$

(*iv*) The squared volatility process $(\sigma_t^2)_{t\geq 0}$ is a time-homogeneous Markovian process. Furthermore, if σ_∞^2 exists and $\sigma_0^2 \stackrel{d}{=} \sigma_\infty^2$ is independent of $(L_t)_{t\geq 0}$, then $(\sigma_t^2)_{t\geq 0}$ is strictly stationary (see Klüppelberg et al. (2004)). The process is stationary if $\sigma_\infty^2 \geq \frac{\beta}{\eta}$ a. s.

(*v*) The bivariate process $(\sigma_t, G_t)_{t\geq 0}$ is a Markovian process. Moreover, if $(\sigma_t^2)_{t\geq 0}$ is stationary and $\sigma_0^2 \stackrel{d}{=} \sigma_\infty^2$, then increments of $(G_t)_{t\geq 0}$ are also stationary (see Klüppelberg et al. (2004)).

1.4.3 Behaviour of sample paths of the COGARCH model

In this section we analyse the behaviour of COGARCH models under parameters changes. In order to evaluate how the model behaves in relation to the variation of parameters, we simulate different sample paths considering that only one parameter change and rest of them keeps constant.

The trajectories have been simulated using the `R` code included in Appendix (A.1). We assume that the variance is bounded as Klüppelberg et al. (2006) proved to guarantee the stationary of the process; in this case, the variance cannot be smaller than β/η.

Most researchers agree that COGARCH models are specially sensitive to parameter φ, which is the parameter responsible of the magnitude of the jumps in the definition of the variance (1.7).

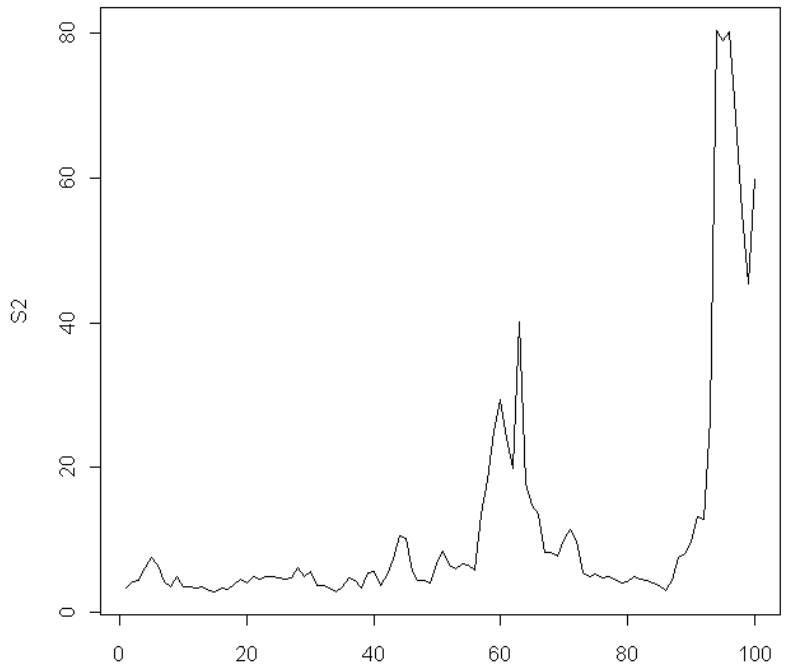

Figure 1.6: Variance of a sample path from a COGARCH(1,1) with parameters $\beta = 0.5$, $\eta = 0.2$ and $\varphi = 0.05$

For instance, in sample paths with high values of φ, the variance presents extremely big jumps and consequently the model has a wild behaviour, that it is not related with real data. As an example, figure 1.6 shows the variance of a sample path with $\varphi = 0.05$.

It may be seen that for $\varphi = 0.05$ the variance can present large values as 80. These large values are actually very large to be real. In fact, the variance in real data depends on the units of assets assuming monetary units (as we did in simulations), therefore it is rare to find variances higher than 5. This behaviour of φ allows us to use an upper bound for this parameter around 0.05.

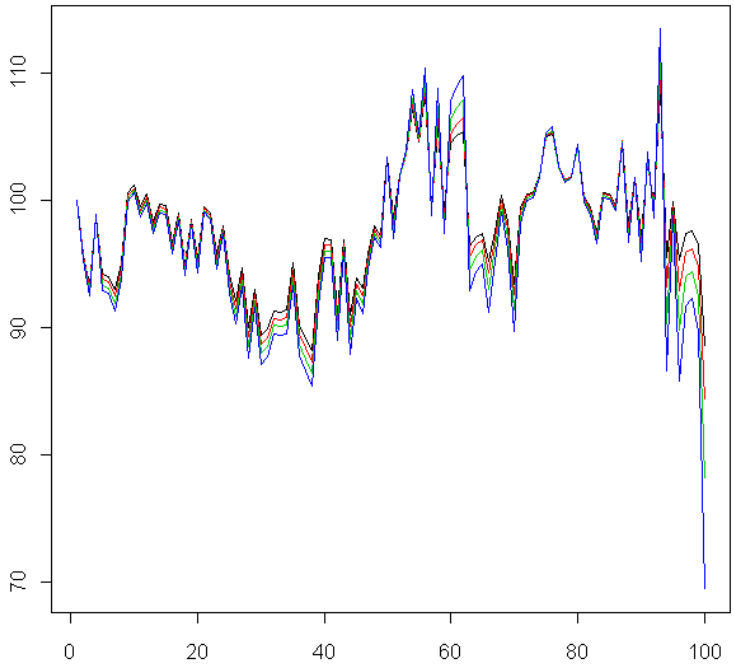

Figure 1.7: Sample paths from a COGARCH(1,1) with parameters $\beta = 0.5$, $\eta = 0.2$ and $\varphi = 0.01,\ 0.02,\ 0.03$ and 0.04.

In order to analyse how the COGARCH(1,1) model adapts to different values of φ, we simulate sample paths for parameters $\eta = 0.2$, $\beta = 0.5$ and $\varphi = 0.01,\ 0.02,\ 0.03$ and 0.04. Figure 1.7 shows the different sample paths that we obtain.

The black line represents the sample path with $\varphi = 0.01$ and the blue one, the sample path with $\varphi = 0.04$. It may be seen that the process presents larger variations along time when φ is also larger, but it is remarkably high just in the jumps of the process. We can interpret this fact assuming that φ is the parameter responsible of these jumps. This fact can be confirmed by the COGARCH variance expression (1.7), where parameter φ is multiplying the quadratic increment of the Lévy process. Figure 1.8 shows variances for the different sample paths.

Variances enable to observe better the effects of changing φ in the model. The

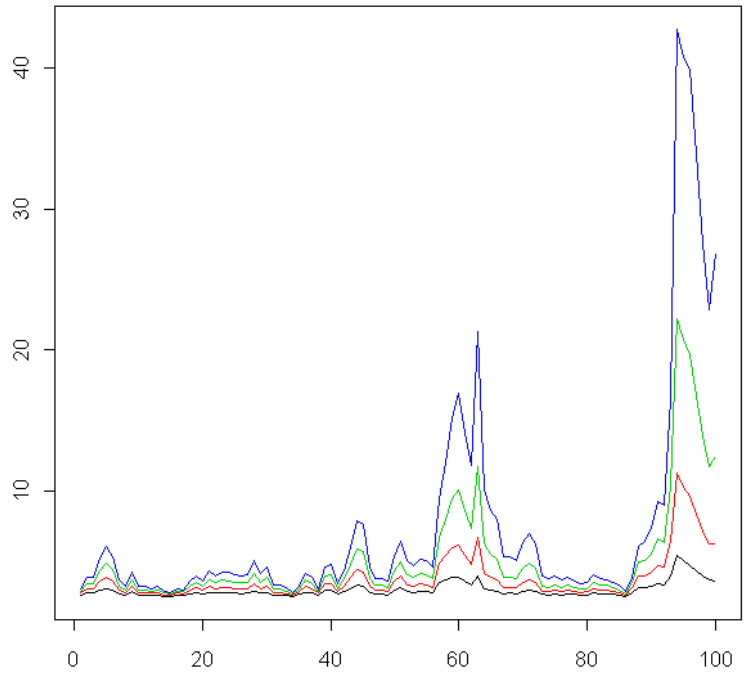

Figure 1.8: Variance of sample paths from a COGARCH(1,1) with parameters $\beta = 0.5$, $eta = 0.2$ and $\varphi = 0.01,\ 0.02,\ 0.03$ and 0.04.

black line represents the sample path with $\varphi = 0.01$ and the blue, the sample path with $\varphi = 0.04$. An exponential growth of the variance may be observed when φ grows.

Müller (2010) also said that this parameter is responsible of the magnitude of the jump in the asset.

Now, we analyse how a COGARCH(1,1) process changes when parameter β varies. Figure 1.9 shows different sample paths from a COGARCH(1,1) model of parameters $\beta = 0.5,\ 1,\ 1.5$ and 2, $\eta = 0.2$ and $\varphi = 0.01$. Line in black corresponds to the smaller value of β, and the larger one is in blue.

Note that as β increases, the COGARCH(1,1) model presents higher variations. But the effect is not the same than when φ increases, as the increment is not exponential; by contrast there is a linear increasing in the variability as

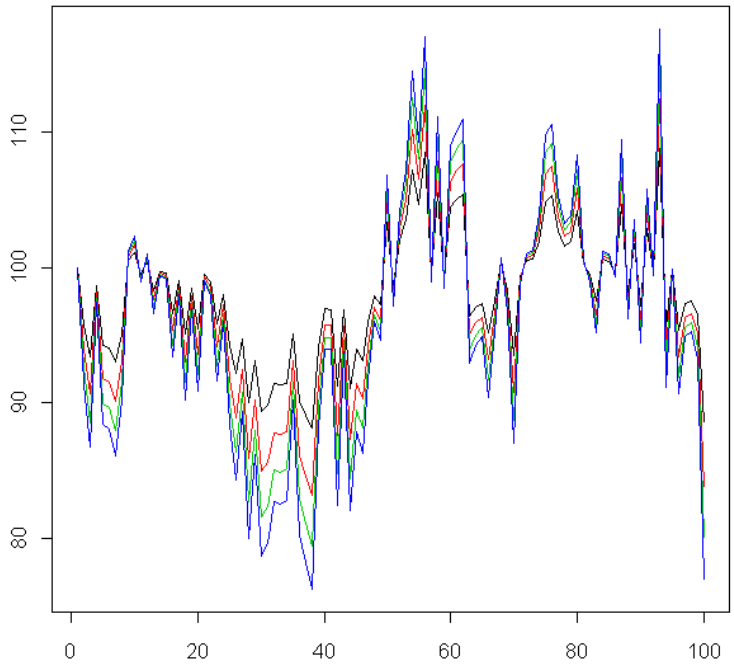

Figure 1.9: Sample path from a COGARCH(1,1) with parameters β=0.5, 1, 1.5 and 2, $\eta = 0.2$ and $\varphi = 0.01$.

β is larger. In figure 1.10 we can observe the behaviour of the variance when β changes.

In the plot about variances (1.10), it can be seen that if β is larger, the lowest bound of the variance is also larger, as expected assuming that the lowest bound is β/η. In the same way, all values of the variance along time are larger.

As Müller (2010) indicated β can be consider as the parameter which measures the overall level of the volatility. Considering this, if β is bigger, this overall value would be higher.

Finally, we show the behaviour of the model in relation with η (see figure 1.11).

Here, the black line corresponds to the smallest η value and blue one, to the largest. It may be seen that, when η increases, the variation of the model

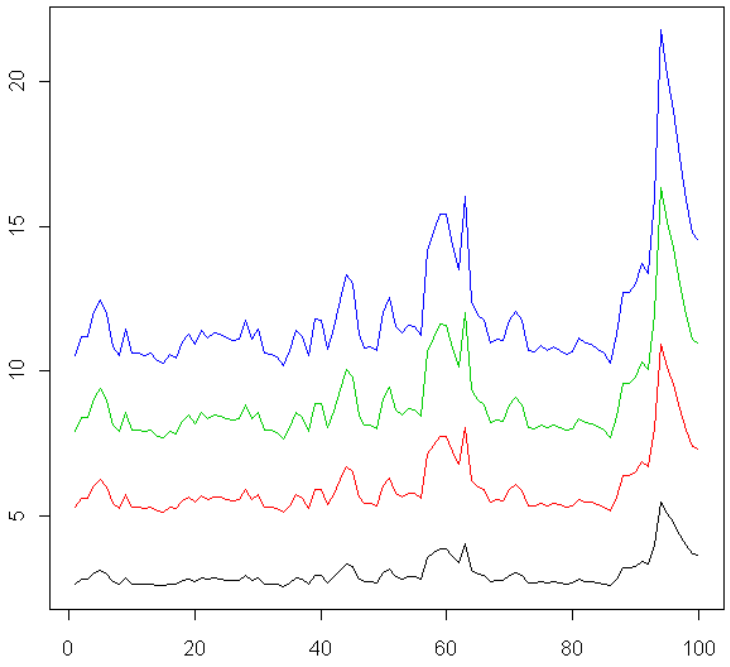

Figure 1.10: Variance of sample paths from a COGARCH(1,1) with parameters $\beta = 0.5,\ 1,\ 1.5$ and 2, $\eta = 0.2$ and $\varphi = 0.01$.

decreases. The behaviour is opposite with respect to β in terms of a linear decreasing (see figure 1.12).

Figure 1.12 shows the relationship between η and the lowest bound of the variance. This was the expected behaviour as the lowest bound expression is (β/η). Furthermore, the variance decreases linearly for all values when η increases.

As the effect of both parameters β and η is quite similar, different combinations of them can provide similar sample paths, which makes the estimation of COGARCH model parameters quite cumbersome. Anyway, this is not an inconvenience when focusing in forecasting tasks.

Müller (2010) interpret this parameter as the responsible of the speed of decline of a volatility burst. According to this, η and φ provide information about how the new information influence the volatility of the asset, or how fast the new

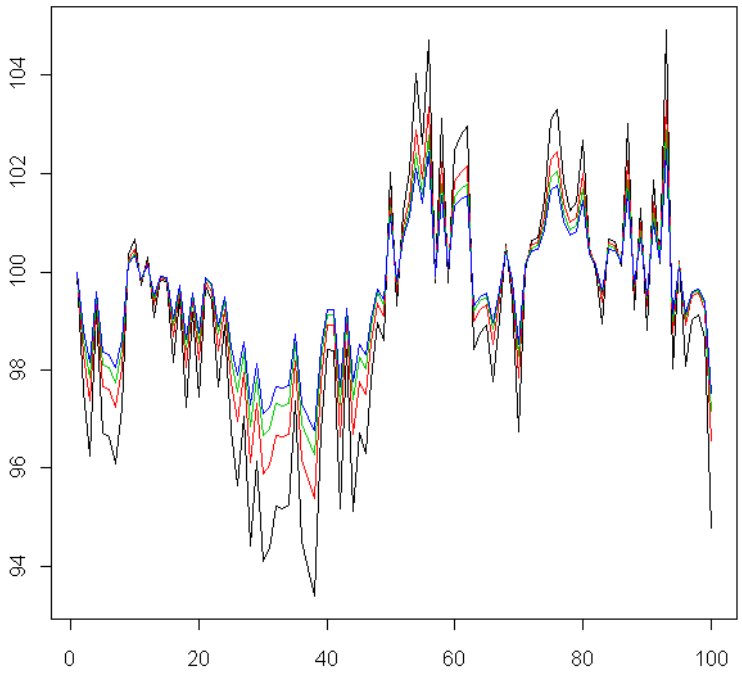

Figure 1.11: Sample path from a COGARCH(1,1) with parameters $\beta = 0.5$, η =0.2 0.6, 1.2 and 2.4 and $\varphi = 0.01$.

information is absorbed by the market.

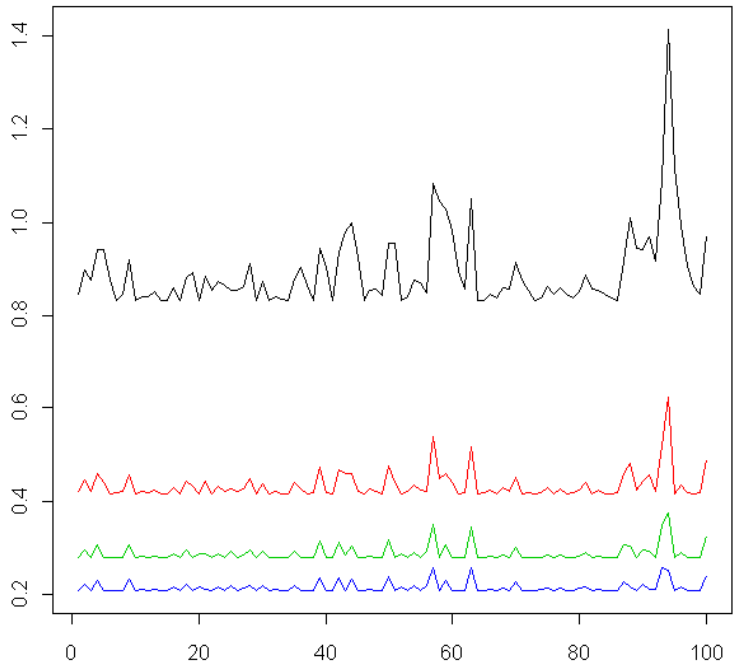

Figure 1.12: Variance of sample paths from a COGARCH(1,1) with parameters $\beta = 0.5$, $\eta = 0.2$, 0.6, 1.2 and 2.4 and $\varphi = 0.01$.

Chapter 2

Estimation of COGARCH models

2.1 Estimation methods in COGARCH models

After the introduction of the COGARCH(1,1) model by Klüppelberg et al. (2004) different authors have dealt with the problem of estimating the parameters of this model. But this is not an easy problem to solve due to the wild behavior of the sample paths of this process, mainly cause by the introduction of a Lèvy process in the model. Among all the solutions, it could be outlined the based on the moments method of Haug et al. (2005), the based on the MCMC framework from Müller (2010) and the based on a pseudo maximum likelihood estimator by Maller et al. (2008). The discretization of the process in the last one and the proposed pseudo likelihood function, will be the base for the solutions we propose to estimate COGARCH model parameters.

2.1.1 Power moments method

The first estimation method was proposed by Haug et al. (2005) and it is based on the power moments method. The estimates of parameters (β, η, φ) are obtained from a sample of equally spaced returns, by matching the empirical autocorrelation function and the power moments to their theoretical equivalents.

In order to obtain the power moments it is necessary to use the Laplace transformation of the auxiliary process defined in (1.5), $E[e^{-sX_t}] = e^{t\Psi(s)}$, where

$$\Psi(s) = -\eta s + \int_{\mathbb{R}} ((1 + \varphi x^2)^s - 1)\nu(dx), \tag{2.1}$$

for $s \geq 0$. The Laplace transformation $E[e^{-sX_t}]$ is finite for one and for all $t > 0$, when $s \geq 0$ is fixed and the previous integral is finite. This is equivalent to the condition $E[L_1]^{2s} < \infty$. The stationarity of the volatility process depends on the existence of a value $s > 0$ which makes $\Psi(s) \leq 0$.

In this way, the power moments for the increments of a COGARCH process G_t, are defined as

$$G_t^{(r)} := G_t - G_{t-r} = \int_{t-r}^{t} \sigma_s dL_s$$

for $t \geq r$, therefore we can define the equidistant sequence of non-overlapped returns of length r, by $\left(G_{nr}^{(r)}\right)_{n\in\mathbb{N}}$.

If the Lévy process $(L_t)_{t\geq 0}$ with finite variance and zero mean, such that $\Psi(1) < 0$, where the process $(\sigma_t^2)_{t\geq 0}$ is stationary. Then, $(G_t)_{t\geq 0}$ has stationary increments $E[G_t^2] < \infty$ for all $t \geq 0$, $0 < r \leq h$ and

$$\begin{aligned} E\left(G_t^{(r)}\right) &= 0, \\ E\left(G_t^{(r)}\right)^2 &= \frac{\beta r}{|\Psi\,(1)|} E[L_1^2], \\ Cov\left(G_t^{(r)}, G_{t+h}^{(r)}\right) &= 0, \end{aligned} \tag{2.2}$$

Furthermore, if $\varphi > 0$, $E[L_1^4] < \infty$, $\Psi(2) < 0$ for all $t \geq 0$, $E[G_t^4] < \infty$ and the Lévy measure ν of L is such that $\int_{\mathbb{R}} x^3 \nu(dx) = 0$, then for every t, $0 < r \leq h$

$$\begin{aligned} E[G_t^{(r)}]^4 &= 6E[L_1^2]\frac{\beta^2}{\Psi(1)^2}\left(2\eta\varphi^{-1} + 2\sigma^2 - E[L_1^2]\right)\left(\frac{2}{|\Psi(2)|} - \frac{1}{|\Psi(1)|}\right) \\ &\quad \times \left(r - \frac{1 - e^{-r|\Psi(1)|}}{|\Psi(1)|}\right) + \frac{2\beta^2}{\varphi^2}\left(\frac{2}{|\Psi(2)|} - \frac{1}{|\Psi(1)|}\right) r \\ &\quad + 3\frac{\beta^2}{\Psi(1)^2}(E[L_1^2])^2 r^2, \end{aligned} \tag{2.3}$$

where σ^2 is the variance of the Brownian motion component of the Lévy process, and

$$\begin{aligned} Cov\left((G_t^{(r)})^2, (G_{t+h}^{(r)})^2\right) &= \frac{\beta^2}{|\psi(1)|^3}\left(2\eta\varphi^{-1} + 2\sigma^2 - E[L_1^2]\right) \\ &\quad \times E[L_1^2]\left(\frac{2}{|\Psi(2)|} - \frac{1}{|\Psi(1)|}\right) \\ &\quad \times \left(1 - e^{-r|\Psi(1)|}\right)\left(e^{r|\Psi(1)|} - 1\right) e^{-h|\Psi(1)|} > 0. \end{aligned} \tag{2.4}$$

Under all the bellow conditions, for each fixed value $r > 0$, the process $\left(\left(G_{nr}^{(r)}\right)^2\right)_{n\in\mathbb{N}}$

has the autocorrelation structure of an ARMA(1,1) process (the proof can be found in Haug et al. (2005)).

Finally, setting $r = 1$, $E[L_1] = 0$ and $E[L_1^2] = 1$ in order to simplify the process, we can obtain the estimators based on the moments method.

Let us consider $\left\{G_n^{(1)}\right\}_{n\in\mathbb{N}}$ the stationary increment process of the COGARCH process with parameters $\beta, \eta, \varphi > 0$ and we denote expressions (2.2), (2.3) and (2.4) in terms of the constants m_1, m_2, k, k_p, p such that

$$\begin{aligned}
E[(G_n^{(1)})^2] &= m_1, \\
E[(G_n^{(1)})^4] &= m_2, \\
p &= |\Psi(1)|, \\
k &= \frac{\beta^2}{|\psi(1)|^3}\left(2\eta\varphi^{-1} + 2\sigma^2 - E[L_1^2]\right) E[L_1^2] \left(\frac{2}{|\Psi(2)|} - \frac{1}{|\Psi(1)|}\right) \\
&\quad \times \left(1 - e^{-r|\Psi(1)|}\right)\left(e^{r|\Psi(1)|} - 1\right), \\
k_p &= \frac{k}{m_2 - m_1^2}, \\
\gamma(h) &= Cov((G_n^{(1)})^2, (G_{n+h}^{(1)})^2) = ke^{-hp}, \\
\rho(h) &= Corr((G_n^{(1)})^2, (G_{n+h}^{(1)})^2) = k_p e^{-hp},
\end{aligned}$$

for $h \in \mathbb{N}$. Now let us define

$$\begin{aligned}
M_1 &:= m_2 - 3m_1^2 - 6\frac{1 - p - e^{-p}}{(1 - e^{p})(1 - e^{-p})}k \\
M_2 &:= \frac{2kp}{M_1(e^p - 1)(1 - e^{-p})}
\end{aligned}$$

Then, when $M_1, M_2 < 0$, the estimates of parameters (β, η, φ) can be obtained by means of

$$\begin{aligned}
\beta &= pm_1, \\
\varphi &= p\sqrt{1 + M_2} - p, \\
\eta &= p + \varphi\left(1 - \sigma^2\right).
\end{aligned}$$

(proofs can be seen in Haug et al. (2005)). In this case, the consistency of the power moments implies the consistency of the estimates of (β, η, φ).

In simulation studies it is shown that this methodology yields good estimations of the COGARCH parameters, but it presents a couple of shortcomings. The first limitation is that it is necessary to consider equidistant observations, although it is not often the case when working with real observations, where changes in prices may be quite different when assuming different periods of time. Actually Mandelbrot (1963) analysed how prices are affected by the long periods of time of closed markets, so it affects estimations. Maller et al. (2008) proposed a discrete time model which may be used with irregularly spaced observations. In this work we will focus mainly on this approach.

Another shortcoming is related with the requirement of having lots of replicates (or simulated data) for obtaining accurate estimators, which of course becomes impossible to achieve in real situations.

2.1.2 Bayesian approach

From a Bayesian point of view, Müller (2010) proposed a methodology for the COGARCH models based on a Markov Chain Monte Carlo (MCMC) approach, assuming that the driven Lévy process is a compound Poisson process susch that $E(L_1) = 0$ and $E(L_1^2) = 1$. They obtained estimations for the main parameters β, η, φ of the model and for parameters related with the process like the intensity of the embedded compound Poisson of the Lévy process (c), the initial variance of the process (σ_0^2) and the market point process ($\boldsymbol{\psi}$).

As the driving Lévy process is a compound Poisson process, the stochastic

differential equation of the variance process of the model (1.7) can be written as

$$d\sigma_t^2 = (\beta - \eta\sigma_{t^-}^2)dt + \varphi\sigma_{t^-}^2(\Delta L_t)^2$$

It is assumed that the second order moment of the jump distribution m_2 is finite, there are no point mass at 0, and we denote the corresponding density function as $z(\cdot)$. The distribution of the inter arrival times of L is exponential, so it is difficult that jumps occur at the observation times. Furthermore, L should also have negative jumps because G is an integrated COGARCH process and its differences are log returns.

The process G is observed in the interval $[0, T]$ and the observation times are $0 < t_1 < \cdots < t_n =: T$. The interval between two consecutive observations is denoted by $I_j := [t_{j-1}, t_j]$ and the increment in G associated to this interval is denoted by $\Delta G_{t_j} := G_{t_j} - G_{t_{j-1}}$.

The estimated jump times of L and consequently of G in $[0, T]$ are $0 < \tau_1 < \cdots < \tau_m < T$, where m is the number of jumps, which can vary for each iteration of the algorithm.

The estimated jump sizes of G are $g_i := \Delta G_{\tau_i}$ for $i = 1, \ldots, m$, and it is set $g_0 := 0$; in this way, g is the vector of the jump sizes, $g := (g_1, \ldots, g_m)$. The time between two consecutive estimated jump times is $\Delta\tau_i := \tau_i - \tau_{i-1}$, and $\tau := (\tau_1, \ldots, \tau_m)$ is the estimated jump times vector. By the other hand, Müller (2010) consider $\psi := \{(\tau_i, g_i), i = 1, \ldots, m\}$ as a marked point process.

Between two consecutive jumps of the Lévy process, the COGARCH process and its volatility do not jump. Furthermore, between two consecutive jumps, the process σ^2 follows an ordinary differential equation (ODE) with the following

exact solution

$$\sigma^2_{\tau_i} = \frac{\beta}{\eta} + \left[\sigma_{\tau^2_{i-1}} + \varphi g^2_{i-1} - \frac{\beta}{\eta}\right] e^{-\eta\Delta\tau_i}. \tag{2.5}$$

As $e^{-\eta\Delta\tau_i} \to 0$, $\lim_{\Delta\tau_i} \sigma^2_{\tau_i} = \frac{\beta}{\eta}$ and this lower bound of volatility may be used in the algorithms.

To find a discrete GARCH equation on the jump times, equation (2.5) can be written as

$$\sigma^2_{\tau_i} = \frac{\beta}{\eta}(1 - e^{-\eta\Delta\tau_i}) + e^{-\eta\Delta\tau_i}\sigma_{\tau^2_{i-1}} + e^{-\eta\Delta\tau_i}\varphi g^2_{i-1}.$$

If θ, g and τ are known or estimated, the volatility $\sigma^2_{\tau_i}$ can be deterministically computed. The MCMC algorithm from Müller (2010) estimates β, η, φ, σ^2_0, c and the market point process ψ, according to the following scheme:

1. Generate the jump times and sizes of G, $\boldsymbol{\psi}$, from $f(\boldsymbol{\psi}|G, \theta, c)$.

2. Generate the COGARCH parameters β, η, φ and σ^2_0 from $f(\theta|G, \boldsymbol{\psi}, \theta)$.

3. Generate c, the intensity of the compound Poisson process L, from $f(c|G, \boldsymbol{\psi}, \theta)$.

The exact computation of the full conditional distributions is not possible, so it is necessary to use a Metropolis-Hasting procedure, assuming this dependence structure

$$f(\boldsymbol{\psi}, \beta, \eta, \varphi, \sigma^2_0, c) = f(\boldsymbol{\psi}|\beta, \eta, \varphi, \sigma^2_0, c) f(\sigma^2_0|\beta, \eta, \varphi, c) f(\varphi|\eta, c) f(\beta) f(\eta) f(c).$$

Müller (2010) used a gamma distribution as the prior distribution for c, a Pareto prior distribution for σ^2_0 (with the proper boundaries in order to guarantee the

stationarity of the process), and uniform distributions for β, η and φ, defined as

$$\begin{aligned} c &\sim \Gamma(a,b) \\ \beta &\sim U(0,\infty) \\ \eta &\sim U(0,\infty) \\ \varphi &\sim U(0,\eta/(cm_2)) \\ \sigma_0^2 &\sim Pareto(\beta/\eta;\eta/(c\varphi m_2)). \end{aligned}$$

In order to define the prior distribution of $\boldsymbol{\psi}$ it should be taken into account that it depends on θ and c. As the inter arrival times of L (equivalently of G), are independent of the jump sizes and parameters, then

$$f(\boldsymbol{\psi}|\theta,c) = f(g|\tau_1,\ldots,\tau_m,m,\theta)f(\tau_1,\ldots,\tau_m|m,c)f(m|c). \tag{2.6}$$

Equation (2.6) shows the decomposition of $f(\boldsymbol{\psi}|\theta,c)$ into the distribution of the number of jumps and the distribution of the position of these jumps. The first factor can be written as

$$f(g|\tau_1,\ldots,\tau_m,m,\theta) = \prod_{i=1}^{m} f(g_i|g_{i-1},\tau_1,\ldots,\tau_m,m,\theta) \tag{2.7}$$

But $f(g_i|g_{i-1},\tau_1,\ldots,\tau_m,m,\theta)$ is equivalent to the distribution of g_i given σ_{τ_i} and the jumps of L have density $z(\cdot)$, so (2.7) can be written as

$$f(g|\tau_1,\ldots,\tau_m,m,\theta) = \prod_{i=1}^{m} \frac{1}{\sigma_{\tau_i}} z\left(\frac{g_i}{\sigma_{\tau_i}}\right).$$

For a fixed number of jumps in $[0,T]$, the position of jumps is uniformly distributed on $0<\tau_1<\cdots<\tau_m<T$ as we are considering a Poisson process; then $f(\tau_1,\ldots,\tau_m|m,c)$ can be written as

$$f(\tau_1,\ldots,\tau_m|m,c) = \frac{m!}{T^m} 1_{\{0<\tau_1<\cdots<\tau_m<T\}}.$$

The probability of having exactly m jumps being c known, is represented by the last factor and assuming that we have a compound Poisson process, it follows

that

$$f(m|c) = e^{-cT}\frac{(cT)^m}{m!}.$$

Therefore, substituting in (2.6),

$$f(\psi|\theta, c) = e^{-cT}c^m \left[\prod_{i=1}^{m} \frac{1}{\sigma_{\tau_i}} z\left(\frac{g_i}{\sigma_{\tau_i}}\right)\right] 1_{\{0<\tau_1<\cdots<\tau_m<T\}}.$$

Note that it is a mixed distribution as it contains discrete and continuous components.

Müller (2010) compared this Bayesian approach with the pseudo-maximum likelihood procedure of Maller et al. (2008), and he concluded that the first one obtained better results from a practical point of view. But, in order to reduce the number of iterations of the MCMC procedure, the starting values of chains in the MCMC procedure were based on the the pseudo-maximum likelihood estimations of the parameters.

2.2 Aproximating COGARCH models

In this section we introduce and summarize the main ideas of COGARCH models for irregularly spaced data proposed by Maller et al. (2008), which will be the basis of the proposal techniques of this work. It is assumed that the driven process is a pure jump Lévy process, that is, a process with no Brownian part ($\sigma^2 = 0$). It is also assumed that $E(L_1) = 0$, $E(L_1^2) = 1$ and that $\mathcal{F}_t$ is the natural filtration of the Lévy process.

Maller et al. (2008) uses a discrete process $G_n = (G_n(t))_{t\geq 0}$, based on a embedded GARCH processes, and he considers an interval $[0, T]$ ($T > 0$) and a sequence of numbers $(N_n)_{n\geq 1}$ such that $\lim_{n\to\infty} N_n = \infty$ and a partition for $[0, T]$ depending on n. Then, for each n, the $[0, T]$ is divided in N_n subintervals with

lengths $\Delta t_i(n) = t_i(n) - t_{i-1}(n)$.

We assume that $\Delta t(n) = \max_{i=1,...,N_n} \Delta t_i(n) \longrightarrow 0$ if $n \to \infty$ and we define for each n a discrete process $(G_{i,n})_{i=1,...,N_n}$

$$G_{i,n} = G_{i-1,n} + \sigma_{i-1,n}\sqrt{\Delta t_i(n)}\epsilon_{i,n},$$

where

$$G_{0,n} = G(0) = 0$$

and

$$\sigma^2_{i,n} = \beta\Delta t_i(n) + (1 + \varphi\Delta t_i(n)\epsilon^2_{i,n})e^{-\eta\Delta t_i(n)}\sigma^2_{i-1,n},$$

for $i = 1, ..., N_n$.

The innovations $\epsilon_{i,n}$ are constructed by using a "first jump" approximation to the Lévy process, defined as

$$\epsilon_{i,n} = \frac{1_{\tau_{i,n}<\infty}\Delta L_{\tau_{i,n}} - E(1_{\tau_{i,n}<\infty}\Delta L_{\tau_{i,n}})}{Var(1_{\tau_{i,n}<\infty}\Delta L_{\tau_{i,n}})}$$

where

$$\tau_{i,n} = \inf\{t \in [t_{i-1}(n), t_i(n)) : |\Delta L\_t| \geq m_n)\}$$

are stopping times for m_n a sequence of strictly positive sequence $1 \geq m_n \downarrow 0$.

On the other hand, we embed the discret time process into a continuous process defined as

$$\begin{aligned} G_n(t) &= G_{i,n} \text{ where } t \in [t_{i-1}(n), t_i(n)] \text{ with } G_n(0) = 0, \\ \sigma_n(t) &= \sigma_{i,n} \text{ where } t \in [t_{i-1}(n), t_i(n)], \end{aligned}$$

And considering the Skorokhod distance ρ (see e.g. Billingsley (2008)) then,

$$\rho((G_n, \sigma_n), (G, \sigma)) \xrightarrow[n\to\infty]{P} 0$$

As the sequence converges in probability, then it also converges in distribution, that is

$$\rho((G_n, \sigma_n), (G, \sigma)) \xrightarrow[n\to\infty]{d} 0$$

In this way, the process $(G_n(t), \sigma_n(t))$ may be considered as an approximation of the COGARCH model $(G(t), \sigma(t))$ in the sense of Klüppelberg et al. (2004), when n is big enough. Therefore, if we consider small enough intervals we can obtain the approximation of the continuous COGARCH process.

Maller et al. (2008) introduced a pseudo-maximum likelihood estimator of the COGARCH parameters, assuming that data are observed in discrete and optionally unequally spaced time points. In this case, Maller et al. (2008) assumed that these time points were fixed; by contrast Müller (2010) assumed in a Bayesian framework, that they were taken at random.

From a practical point of view, daily financial time series are observed on unequally spaced time points, but these are not taken at random because they depend on weekends and fixed holidays. Furthermore these time points are not difficult to predict so make sense consider them as known information.

It is assumed that $G(t_i)$ are taken from a COGARCH(1,1) model, in time points $0 = t_0 < t_1 < \cdots < t_N = T$. Then, the observed returns are $Y_i = G(t_i) - G(t_{i-1})$ with time points increments $\Delta t_i = t_i - t_{i-1}$.

Y_i are conditionally independent given the previous returns, $\mathcal{F}_{t_{i-1}}$. Furthermore, Y_i conditional on $\mathcal{F}_{t_{i-1}}$ follows a normal distribution $N(0, \rho_i^2)$ where

$$\rho_i^2 = Var(Y_i|\mathcal{F}_{t_{i-1}}) = E(Y_i^2|\mathcal{F}_{t_{i-1}}) = \left(\sigma_{i-1}^2 - \frac{\beta}{\eta - \varphi}\right)\left(\frac{e^{(\eta-\varphi)\Delta t_i} - 1}{\eta - \varphi}\right) + \frac{\beta \Delta t_i}{\eta - \varphi} \tag{2.8}$$

and

$$\begin{aligned}
\sigma_i^2 &= \beta\Delta t_i + e^{-\eta\Delta t_i}\sigma_{i-1}^2 + \varphi e^{-\eta\Delta t_i}\Delta t_i \epsilon_{i,n}^2 \sigma_{i-1}^2 \\
&= \beta\Delta t_i + e^{-\eta\Delta t_i}\sigma_{i-1}^2 + \varphi e^{-\eta\Delta t_i}(\sqrt{\Delta t_i}\epsilon_{i,n}\sigma_{i-1})^2 \\
&= \beta\Delta t_i + e^{-\eta\Delta t_i}\sigma_{i-1}^2 + \varphi e^{-\eta\Delta t_i}(G(t_i) - G(t_{i-1}))^2 \\
&= \beta\Delta t_i + e^{-\eta\Delta t_i}\sigma_{i-1}^2 + \varphi e^{-\eta\Delta t_i} Y_i^2.
\end{aligned}$$

To assure the variance stationary of the process it is necessary that $E[\sigma_0^2] = \frac{\beta}{\eta-\varphi}$. Then, as starting value, it is taken $\sigma_0^2 = \frac{\beta}{\eta-\varphi}$ being $\eta > \varphi$).

Finally, the pseudo-log-likelihood function for the returns $Y_1, Y_2, \ldots, Y_N$ can be expressed as

$$L(\beta, \varphi, \eta) = \left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\rho_i^2}}\right) \exp\left\{-\sum_{i=1}^{N} \frac{Y_i^2}{2\rho_i^2}\right\}. \tag{2.9}$$

Based on the previous expression, Maller et al. (2008) proposed an algorithm to maximize this pseudo-log-likelihood function and he obtained the estimates values for (β, η, φ). In comparison with the power moment methods of Haug et al. (2005), the estimates were more accurate, although they were not consistent.

In the next section we introduce a simple and straightforward MCMC estimator for COGARCH(1,1) model parameters, based on Maller et al. (2008) approach.

2.3 A Bayesian approach to COGARCH models

In this work we propose a simple Bayesian approach to analyse and estimate the parameters of the COGARCH(1,1) model, based on the parametrization of Klüppelberg et al. (2004) and Maller et al. (2008). We assume that time points are not taken at random, then we will determine the posterior distributions of the parameters, without simulating the time points in the sense of Müller (2010).

Before defining the Bayesian estimator it is interesting remark the condition to guarantee the existence of a stationary distribution for the volatility process (see Klüppelberg et al. (2006)). It will be used the auxiliary process $(X_t)_{t\geq 0}$ defined by (1.5), the Laplace transform $E[e^{-sX_t}] = e^{t\Psi(s)}$ and the Laplace exponent $\Psi(s)$ from (2.1).

If it exists some $s > 0$ such that $\Psi(s) \leq 0$ then it will exist a stationary distribution for the variance process σ^2. Furthermore, if $\Psi(s) < 0$ for some $s > 0$ then $\Psi(t) < 0$ for all $0 < t \leq s$. It is difficult to find an explicit condition on model parameters η, φ and the Lévy process parameters which makes $\Psi(s) < 0$ for any $s > 0$. But setting s=1 the Laplace exponent can be write as

$$\Psi(1) = -\eta + \varphi \int_{\mathbb{R}} x^2 \nu(dx)$$

Assuming that L has not Brownian motion component and it is a compound Poisson process with intensity c and a jump distribution with the second moment m_2 finite, it can be obtained that $\Psi(1) = c\varphi m_2 - \eta$. Considering that the condition to guarantee the stationarity of the distribution for the volatility process is

$$c\varphi m_2 - \eta \leq 0 \tag{2.10}$$

It is possible to derive an equivalent condition for certain jump distribution and some $s < 1$, which could be considered a weaker condition to be used instead of (2.10). Furthermore, Klüppelberg et al. (2004) proved that $E[\sigma^{2k}]$, with $k \in \mathbb{N}$ exists, if and only if $E[L_1^{2k}] < \infty$ and $\Psi(k) < 0$. In this situation for $k = 1, 2$ we can obtain

$$E[\sigma_t^2] = \frac{\beta}{|\Psi(1)|} \quad and \quad E[\sigma_t^4] = \frac{2\beta^2}{|\Psi(1)\Psi(2)|}. \tag{2.11}$$

Once we are able to guarantee the stationarity of our process we are ready to introduce our estimator. We consider for parameters (β,η,φ) uniform prior distributions in order to deal with weakly informative distributions. All parameters

must be positive, $\beta > 0$, $\eta > 0$, $\varphi \geq 0$, and we will assume that the corresponding Lévy process is determined with $c = 1$ and $m_2 = 1$. Therefore, the previous stationarity condition (2.10) is simplified as $\varphi < \eta$.

Note that φ is the parameter responsible of the process jumps and it multiplies the quadratic increment of the Lévy process in the definition of the variance (1.7). COGARCH models are quite sensible to changes in the values of parameters, and it can be shown in simulation studies that it is especially sensible to changes in φ as this parameter is the responsible for the process jumps.

When setting $c = 1$ and $m_2 = 1$, φ must be less than 0.05 in order to model the real behaviour of financial series. Larger values of φ make the process to have unrealistic big jumps. As example of the magnitude of the jumps which may be observed just in only one day, if φ is set as larger than 0.05, returns could increase or decrease the stock price about 20000%, which is impossible in real data. Even if $\varphi = 0.05$ jumps may increase or decrease of the stock price around 200% in just one day, with is actually very high form a practical point of view. But to guarantee the good behaviour of our estimator it is not necessary setting a smaller valuer of φ.

Consequently we set uninformative uniform prior distributions for the parameters of the model

$$\begin{aligned} \beta &\sim U(0, a), \\ \eta &\sim U(\varphi, b), \\ \varphi &\sim U(0, c), \end{aligned}$$

where a, b and c are positive real values and $b > c$.

By using the pseudo-log-maximum distribution introduced by Maller et al.

(2008) the joint posterior distribution of the parameters can be defined as

$$\begin{aligned}\pi(\beta,\varphi,\eta|Y_1,\ldots,Y_N) &\propto L(\beta,\varphi,\eta)\pi(\beta)\pi(\varphi)\pi(\eta)\\ &\propto \left(\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi\rho_i^2}}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2\rho_i^2}\right\}\\ &\propto \left(\prod_{i=1}^{N}\frac{1}{\rho_i}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2\rho_i^2}\right\}.\end{aligned}$$

and the conditional distribution of the parameters as

$$\begin{aligned}\pi(\beta|\varphi,\eta,Y_1,\ldots,Y_N) &\propto \left(\prod_{i=1}^{N}\frac{1}{\rho_i}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2\rho_i^2}\right\},\\ \pi(\varphi|\beta,\eta,Y_1,\ldots,Y_N) &\propto \left(\prod_{i=1}^{N}\frac{1}{\rho_i}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2\rho_i^2}\right\},\\ \pi(\eta|\beta,\varphi,Y_1,\ldots,Y_N) &\propto \left(\prod_{i=1}^{N}\frac{1}{\rho_i}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2\rho_i^2}\right\}.\end{aligned}$$

The algorithm starts with the initial solution $\beta^{(0)}$, $\varphi^{(0)}$, $\eta^{(0)}$ and, in each iteration (m) generate values for $\beta^{(m)}$, $\varphi^{(m)}$, $\eta^{(m)}$ as follows

Step 1: Set the initial values $\beta^{(0)}$, $\varphi^{(0)}$, $\eta^{(0)}$, and set $m = 1$.

Step 2: Generate the value of $\beta^{(m)}$ by

$$\begin{aligned}\beta^{(m)} &\propto \left(\prod_{i=1}^{N}\frac{1}{\rho_{i,\beta}^{(m-1)}}\right)\exp\left\{-\sum_{i=1}^{N}\frac{Y_i^2}{2(\rho_{i,\beta}^{(m-1)})^2}\right\}\\ \rho_{i,\beta}^{(m-1)} &= \sqrt{\Bigg(\quad\left(\sigma_{i-1,\beta}^2-\frac{\beta}{\eta^{(m-1)}-\varphi^{(m-1)}}\right)\left(\frac{\exp\{(\eta^{(m-1)}-\varphi^{(m-1)})\Delta t_i\}-1}{\eta^{(m-1)}-\varphi^{(m-1)}}\right)\\ &\quad +\frac{\beta\Delta t_i}{\eta^{(m-1)}-\varphi^{(m-1)}}\Bigg)}\\ \sigma_{i-1,\beta}^2 &= \beta\Delta t_{i-1}+\exp\{-\eta^{(m-1)}\Delta t_{i-1}\}\sigma_{i-2,\beta}^2+\varphi^{(m-1)}\exp\{-\eta^{(m-1)}\Delta t_{i-1}\}Y_{i-1}^2\\ \sigma_{0,\beta}^2 &= \frac{\beta}{\eta^{(m-1)}-\varphi^{(m-1)}}\end{aligned}$$

Step 3: Generate the value for $\varphi^{(m)}$ using $\beta^{(m)}$ in

$$\begin{aligned}
\varphi^{(m)} &\propto \left(\prod_{i=1}^{N} \frac{1}{\rho_{i,\varphi}^{(m-1)}}\right) \exp\left\{-\sum_{i=1}^{N} \frac{Y_i^2}{2(\rho_{i,\varphi}^{(m-1)})^2}\right\} \\
\rho_{i,\varphi}^{(m-1)} &= \sqrt{\left(\sigma_{i-1,\varphi}^2 - \frac{\beta^{(m)}}{\eta^{(m-1)} - \varphi}\right)\left(\frac{\exp\{(\eta^{(m-1)} - \varphi)\Delta t_i\} - 1}{\eta^{(m-1)} - \varphi}\right) + \frac{\beta^{(m)}\Delta t_i}{\eta^{(m-1)} - \varphi}} \\
\sigma_{i-1,\varphi}^2 &= \beta^{(m)}\Delta t_{i-1} + \exp\{-\eta^{(m-1)}\Delta t_{i-1}\}\sigma_{i-2,\beta}^2 + \varphi \exp\{-\eta^{(m-1)}\Delta t_{i-1}\}Y_{i-1}^2 \\
\sigma_{0,\varphi}^2 &= \frac{\beta^{(m)}}{\eta^{(m-1)} - \varphi}
\end{aligned}$$

Step 4: Generate the value for $\eta^{(m)}$ by using $\beta^{(m)}$ and $\varphi^{(m)}$ in

$$\begin{aligned}
\eta^{(m)} &\propto \left(\prod_{i=1}^{N} \frac{1}{\rho_{i,\eta}^{(m-1)}}\right) \exp\left\{-\sum_{i=1}^{N} \frac{Y_i^2}{2(\rho_{i,\eta}^{(m-1)})^2}\right\} \\
\rho_{i,\eta}^{(m-1)} &= \sqrt{\left(\sigma_{i-1,\eta}^2 - \frac{\beta^{(m)}}{\eta - \varphi^{(m)}}\right)\left(\frac{\exp\{(\eta - \varphi^{(m)})\Delta t_i\} - 1}{\eta - \varphi^{(m)}}\right) + \frac{\beta^{(m)}\Delta t_i}{\eta - \varphi^{(m)}}} \\
\sigma_{i-1,\eta}^2 &= \beta^{(m)}\Delta t_{i-1} + \exp\{-\eta\Delta t_{i-1}\}\sigma_{i-2,\beta}^2 + \varphi^{(m)} \exp\{-\eta\Delta t_{i-1}\}Y_{i-1}^2 \\
\sigma_{0,\eta}^2 &= \frac{\beta^{(m)}}{\eta - \varphi^{(m)}}
\end{aligned}$$

Step 5 Accept or reject the proposed values $\beta^{(m)}$, $\eta^{(m)}$ and $\varphi^{(m)}$ according to the Metropolis Hasting updates. Set $m = m + 1$ and go to Step 2.

The MCMC algorithm to estimate the COGARCH model parameters has been programmed using Jags (Plummer et al. (2003)) software by means of the package runjags (Denwood (2011)) from the R project (R Core Team (2012)). Code is available in appendix A.2

This MCMC algorithm is different from the proposed by Müller (2010) in different ways. On one hand he estimates the model parameters and the jump times, but, as we base our algorithm in the approximation of Maller et al. (2008), we consider the observed times and focus in the estimation of model parameters. On the other hand we estimate the parameters $\beta^{(m)}$, $\varphi^{(m)}$, $\eta^{(m)}$ in each iteration

and accept or reject them according to Metropolis Hasting algorithm, but he estimate also σ_0^2 and c (the intensity of the Poisson process which drive both, model and variance). Furthermore, even though Müller (2010) algorithm is based on the the Metropolis-Hasting accept or reject structure, he applied a special "Birth moves" and "Death moves" to increase decrease the number of jumps which complicate the algorithm.

2.4 Simulation study about the Bayesian approach to COGARCH models

To assess the quality of the Bayesian approach of section 2.3 we have undertaken a simulation study for a COGARCH(1,1) model. We have fixed the parameters as $\beta = 0.25$, $\eta = 0.35$ and $\varphi = 0.02$ (see the corresponding code in `R` in Appendix A.2.

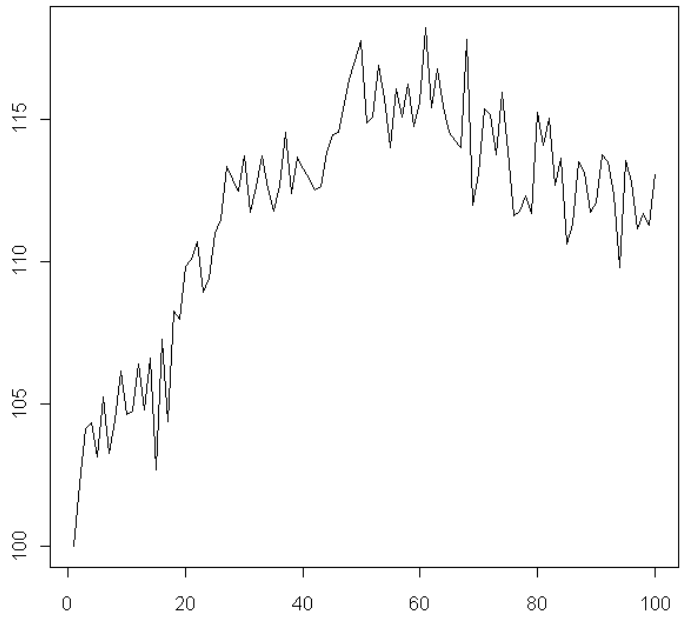

Figure 2.1: COGARCH sample path simulation

Simulated data path is shown in figure 2.1. In this figure it may be observed that simulated sample path shows a similar behaviour than real financial series. See, for instance figure 2.1 compared with real data 1.1. Although, we will focus basically on returns.

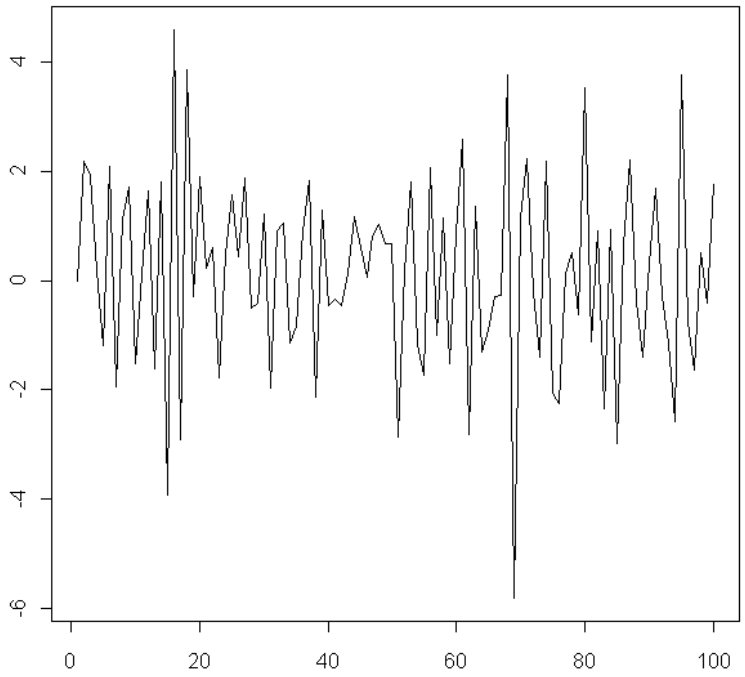

Figure 2.2: Returns of COGARCH sample path simulation

Figure 2.2 shows the returns of a simulated COGARCH model sample path. See the similar shape as real returns from Ibex-35 between 2009 and 2014 (1.3), although, in this case, the simulated COGARCH returns present a higher variability. Therefore, if we may estimate the COGARCH parameters from a process with high variability, hopefully we will obtain better results for real data with smaller variability.

Figure 2.3 shows the variance of a simulated COGARCH process. This variance may be also compared with the variance of a real financial series, like the showed in figure (1.2). Again can be observed that both figures present a similar

picture.

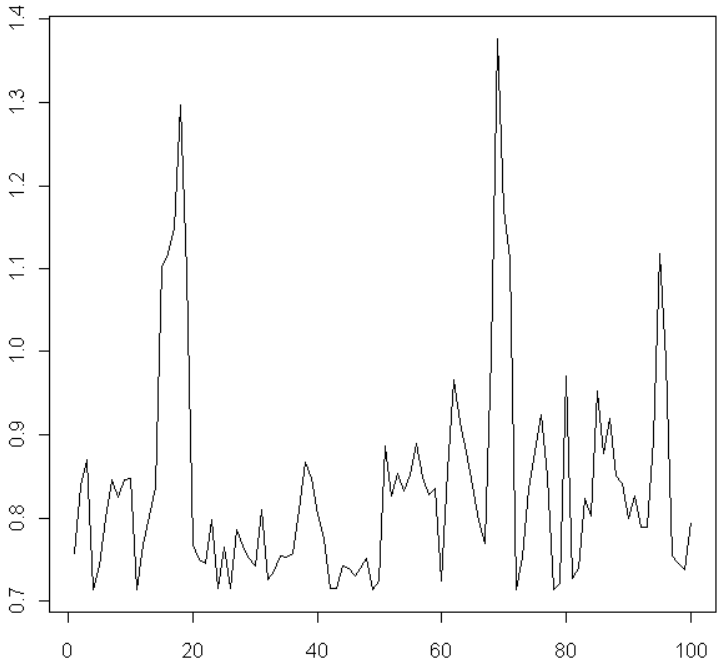

Figure 2.3: COGARCH Variance

We have applied the Bayesian approach introduced in section 2.3 in the simulated data in order to obtain the estimated values for β, η and φ parameters (see the corresponding R code in A.2). A resume of estimation is shown in table 2.1.

The real values of β, η and φ parameters, posterior means, standard deviations and HPD intervals are shown from the second to the fifth columns. The best estimations are obtained for φ and η parameters, although for β the estimate is worst. The corresponding standard deviations are not large. Anyway, the real value of all parameters are included inside the HPD intervals. It can be noted that accurate results are obtained from a single simulated sample path from a COGARCH(1,1) model, in contrast with Maller et al. (2008) or Müller (2010) where they use several sample paths. This is an important advantage, as in real data, time series just consist in a single path. By the other hand, the initial values

Parameter	Real Value	Posterior means	S.D.	HPD 0.95
β	0.25	0.427	0.158	(0.138 ; 0.747)
η	0.35	0.301	0.132	(0.092 ; 0.600)
φ	0.02	0.029	0.014	(0.002 ; 0.049)

Table 2.1: Estimation for COGARCH parameters using MCMC

for the corresponding chain of the MCMC procedure are taken at random without needing to take as starting points the pseudo-maximum likelihood estimators as in Müller (2010).

Anyway, we have also studied the case with more than one trajectory, simulating $m = 500$ sample paths from a COGARCH(1,1) model (see the `R` code included in Appendix A.3).

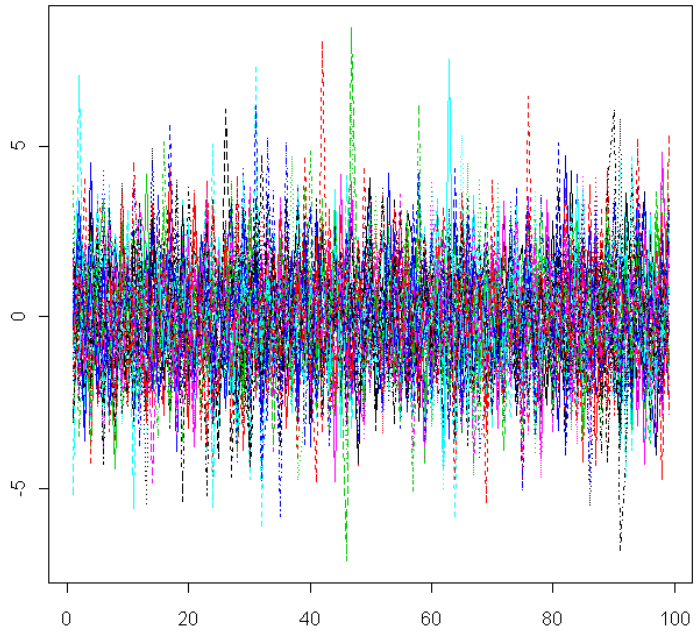

Figure 2.4: Simulated 50 trajectories of a COGARCH(1,1) model

In figure 2.4 it is shown the increments of 50 simulated sample paths of a COGARCH(1,1) model. It can be noted a certain wild behaviour of the processes, although all the trajectories are located inside a similar range of values and they

present common shapes.

The real values of β, η and φ parameters, posterior means, standard deviations and HPD intervals are shown from the second to the fifth columns in table 2.2. When comparing table 2.1 and table 2.2 it can be noticed that there are little

Parameter	Real Value	Posterior means	S.D.	HPD Intervals
β	0.25	0.473	0.154	(0.141,0.860)
η	0.35	0.291	0.073	(0.083,0.589)
φ	0.02	0.031	0.003	(0.004, 0.049)

Table 2.2: Estimates of the COGARCH(1,1) parameters using multiple trajectories

improvements in estimates of β and η by using 50 trajectories instead of only one. As expected, the standard errors in the case of 50 replicates are smaller than in the case of one trajectory. We can conclude that with one set of data there is enough information for finding out accurate estimations, in a COGARCH(1,1), with this Bayesian procedure.

2.5 Real data application of the Bayesian approach to COGARCH models

Finally, in this section, we apply the Bayesian approach of section 2.3 to real data. We use daily observations of the exchange rate between the European currency and the USA Dollar. Data have been obtained from the website `www.infomercados.com` which offers the complete information about this exchange rate from 1993. We analyse the corresponding series between 2011 and 2012.

Figure 2.5 shows the evolution of the exchange rate EURO-USD from January, 3^{rd} of 2011 until November, 5^{th} of 2012. By the other hand, figure 2.6 shows the

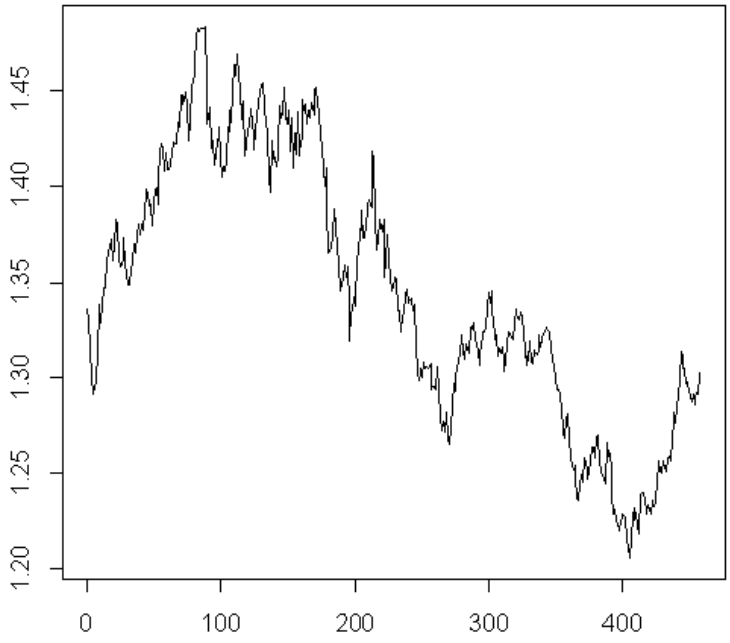

Figure 2.5: Evolution of Euro Dollar exchange rate in 2011 and 2012

daily returns, calculated as the logarithm one lag returns $\log\left(\frac{G_t}{G_{t-1}}\right)$ (see e.g. Francq and Zakoian (2010)).

Table 2.3 shows the results of posterior means, standard deviations and HPD intervals from the second to the forth columns, when using the returns of Euro-USD evolution data.

Parameter	Posterior means	S.D.	HPD Interval
β	1.7051e-06	1.423001e-06	(4.0555e-07, 6.3176e-06)
η	0.0748	4.472574e-02	(0.0300, 0.2197)
φ	0.0348	9.554494e-03	(0.0150, 0.0492)

Table 2.3: Estimates for the returns of Euro-USD evolution data in 2011-2012

With these values for parameters, COGARCH(1,1) model can be used to model Euro-USD exchange rate series, arising new information of the relation between these two currencies. It can also be used to predict future values of this series.

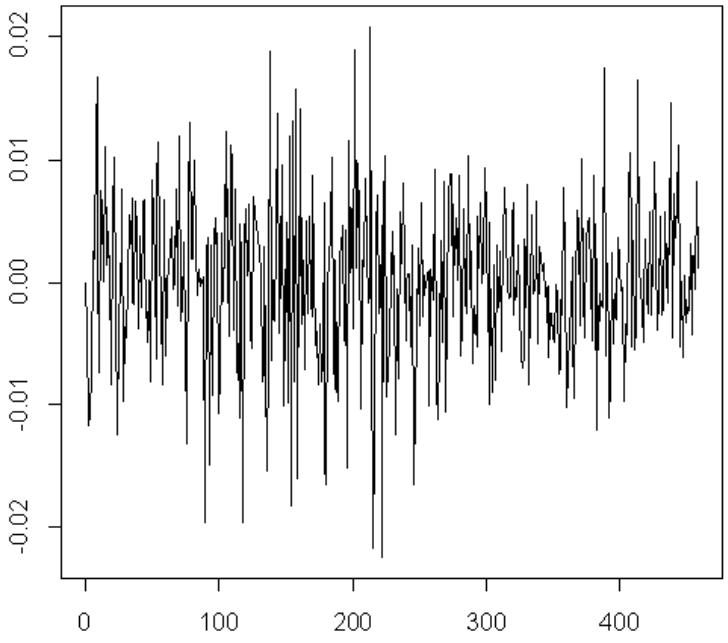

Figure 2.6: Euro Dollar returns in 2011 and 2012

Parameters η and φ represent how the value assume the new information of the markets. η measures how fast the price incorporate the news which affect directly the assets (in this case the price of the exchange rate). φ measures the magnitude of the jumps or volatility burst arising after the arrival of new announcements, which in the model are represented by the jumps of the Lévy. We can notice that a value of 0,03 for φ lets the exchange rate to have large jumps, so it can be said that EURO-DOLAR price is able to have large movements up or down.

On the other hand, β represents the level of volatility overall, and in this example it takes a value of 1.7051e-6, so even though big volatility jumps can be found for this exchange rate, the usual level of volatility is small.

Chapter 3

Data cloning Estimation for GARCH and COGARCH model parameters

3.1 Introduction to data cloning theory

In this chapter new estimators for the models we have considered in this thesis will be introduced, now applying Data Cloning methodology.

In many complex models, as COGARCH model, it is not possible to obtain estimators from a frequentist point of view, but by means of hierarchical Bayesian models it turns out possible to deal with this class of models. Nevertheless, it is necessary to determine the prior distributions for each of the parameters, and sometimes in many fields this fact turns to be a controversial issue.

The interpretation of credible intervals is sometimes another point of discussion. According to a frequentist point of view, these kind of intervals are meaningless as they are not objective because they are strongly based on the beliefs of the analyst.

In the situation of complex models, as using likelihood-based inference is many times difficult to implement, the alternative of data cloning provides an elegant solution. This methodology obtains likelihood-based estimations and frequentist confidence intervals, by means of a basic bayesian methodology.

Initially, data cloning techniques have been used in hierarchical models related with Ecology, where these kind of models are common to deal with related data structures. In this chapter we will use data cloning to estimate the parameters of GARCH and COGARCH models.

Data cloning theory was introduced by Lele et al. (2007) and it is basically a computing methodology which calculates maximum likelihood estimations, using MCMC methods in models whose complexity makes necessary the use of high-dimensional integration to obtain estimates of parameters. By the other hand, it

can be easy implemented by using standard Bayesian software such as `WinBugs` or `Jags`.

The basic idea of the data cloning methodology, is supposing that an experiment can be repeated several times and in all of them the same data are obtained. In our context we can consider that we obtain several sample paths from the same GARCH and COGARCH models, and they all result to have exactly the same values in any time. It is not necessary that they are independent between them.

In this situation, the likelihood function is $L(\theta, y_t^{(k)}) = [L(\theta, y_t)]^{(k)}$ for each replicate k and the maximum of $L\left(\theta, y_t^{(k)}\right)$ is equal to the maximum of $L(\theta, y_t)$. Furthermore, the Fisher information matrix of $L\left(\theta, y_t^{(k)}\right)$ is k times the Fisher information matrix of $L(\theta, y_t)$.

The MCMC methodology is used to calculate the posterior distributions for the parameters of the model. And, as usual in Bayesian theory, this distribution includes the information of data, but now data is not just one sample path but several ones with the same results, namely k clones. However, the efficiency of the estimators is not improved by the artificial increase of the sample size, but the accuracy of estimators as it is remarked in Lele et al. (2010).

By the other hand, proper prior distributions of parameters must be set and, in order to obtain the posterior distribution, an MCMC algorithm is used in the standard way but over the k clones. In such a way, the mean of the posterior distribution approximates the maximum likelihood estimators (see Lele et al. (2007)) and the posterior variance multiplied by the number of clones, approximates the asymptotic variance of the maximum likelihood estimator.

The process can be fully described as follows. We consider n observations,

$Y = (Y_1, Y_2, \ldots, Y_n)$ which can be obtained from a hierarchical model such as

$$\begin{aligned} Y &\sim f(y|X, \varphi) \\ X &\sim g(x|\theta) \end{aligned}$$

where

- X is a vector of random variables or processes which affect the observations.
- f and g are the joint density functions of both random variables.
- $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_n)$ is a vector which contains the parameters related with Y, which defines the model.
- $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ is a vector which contains the parameters related with X.

For this model, the likelihood function is

$$L(\theta, \varphi; y) = \int f(y|X, \varphi) g(X|\theta) dX,$$

where y is the observed data from the random variable Y.

In order to obtain the frequentist maximum likelihood estimators for this model it is necessary to obtain the values of $(\theta, \varphi) = (\theta_1, \theta_2, \ldots, \theta_n, \varphi_1, \varphi_2, \ldots, \varphi_n)$ which jointly maximize the likelihood function, but this is not an easy to solve problem with complex or hierarchical models. Calculating these estimators involves to use high-dimensional integration, and computation of standard errors requires to calculate second derivatives of the log-likelihood function. On the contrary, a Bayesian approach does not require neither finding the maximum of functions nor computing high-dimensional integrals.

If the prior distributions of the parameters are denoted by $\pi(\theta, \varphi)$, then the joint conditional distribution is

$$h(\theta, \varphi, X|y) = \frac{f(y|X, \varphi)g(X|\theta)\pi(\theta, \varphi)}{\int f(y|X, \varphi)g(X|\theta)\pi(\theta, \varphi)dXd\theta d\varphi} \propto f(y|X, \varphi)g(X|\theta)\pi(\theta, \varphi)$$

and the marginal conditional distributions of the parameters, $\pi(\theta, \varphi|y)$, are easily obtained by using MCMC methods.

The data cloning algorithm can be summarized in three steps

Step 1: Create a k-cloned data set $\mathbf{y}^{(k)} = (\mathbf{y}, \mathbf{y}, \ldots, \mathbf{y})$, where the observed data vector $\mathbf{y}$ is repeated k times.

Step 2: Using an MCMC algorithm, generate random observations from the posterior distribution of the parameters that is based on the prior distribution $\pi(\beta, \varphi, \eta)$ and the cloned data vector $\mathbf{y}^{(k)} = (\mathbf{y}, \mathbf{y}, \ldots, \mathbf{y})$, where the k copies of $\mathbf{y}$ are assumed to be independent of each other. In practice, any proper prior distribution can be used.

Step 3: Compute means and variances of the samples values θ_j $(j = 1, \ldots, B)$, for B iterations of the MCMC run generated from the marginal posterior distributions. The posterior mean values correspond to the *ML* estimates of θ_j and k times the posterior variances correspond to the approximate variances of the *ML* estimates.

It can be noted that this algorithm could be used to estimate GARCH model parameters, but it could be specially interesting when estimating COGARCH model parameters, considering that maximum likelihood estimations can be obtained for the first, but not for the second.

3.2 Data cloning estimator for GARCH model parameters

As GARCH models are broadly extended models in time series analysis with no constant variance, in this section we apply the data cloning methodology to calculate maximum likelihood estimators for them.

After that, the quality of the obtained estimator will be evaluated over simulated and real sample paths.

We refer to section 1.3 for the definition of a GARCH(1,1) model,

$$\begin{aligned} Y_n &= \sigma_n \epsilon_n, \\ \sigma_n^2 &= \beta + \lambda Y_{n-1}^2 + \delta \sigma_{n-1}^2, \end{aligned}$$

for $n \geq 0$ where $\beta > 0$, $\lambda \geq 0$ and $\delta \geq 0$, and ϵ_n are a sequence of random variables, independently and identically normally distributed.

In this section we apply the data cloning methodology to approximate the maximum likelihood estimators for β, λ and δ.

In the first step we set the prior distributions of the parameters. In data cloning methods (see Lele et al. (2007)) estimates do not depend heavily on the prior distributions, but we must set the restrictions about them.

In GARCH models $\beta > 0$, $\lambda \geq 0$ and $\delta \geq 0$, so prior distributions only are proposed for positive values. Furthermore, it is necessary to assume more restrictions, as it is shown in proposition 1.

Proposition 9. *If a GARCH(1,1) process satisfies that* $E\left[\log(\lambda \epsilon_n^2 + \delta)\right] < 0$*, then the process is strictly stationary and* λ *and* δ *are bounded between the following*

limits

$$\delta \in (0,1) \tag{3.1}$$

$$\lambda \in (0, 2e^{\gamma}) \tag{3.2}$$

Proof. As $E\left[\log(\lambda\epsilon_n^2 + \delta)\right] < 0$ and $\lambda \geq 0$

$$E\left[\log(\delta)\right] \leq E\left[\log(\lambda\epsilon_n^2 + \delta)\right] < 0$$

that is, $\delta \in (0,1)$, as in (3.1).

In order to obtain (3.2), as $\delta \geq 0$

$$E\left[\log(\lambda\epsilon_n^2)\right] \leq E\left[\log(\lambda\epsilon_n^2 + \delta)\right] < 0 \tag{3.3}$$

As $\epsilon_n \sim N(0,1)$, the characteristic function of $Y = \log(\epsilon_n^2)$ is

$$\begin{aligned}
\varphi_Y(n) &= E\left[e^{inY}\right] = E\left[e^{in\log(\epsilon_n^2)}\right] = E\left[(\epsilon_n^2)^{in}\right] \\
&= \int_0^{\infty} \frac{1}{\sqrt{2\pi}} \left(\epsilon_n^2\right)^{in-1/2} e^{-\epsilon_n^2/2} d\epsilon_n^2 \\
&= \frac{1}{\sqrt{2\pi}} \frac{\Gamma\left(\frac{2in+1}{2}\right)}{(1/2)^{\frac{2in+1}{2}}}
\end{aligned}$$

which gives

$$E\left[\log\left(\epsilon_n^2\right)\right] = \frac{\varphi_Y'(0)}{i} = \frac{\Gamma'(1/2)}{\sqrt{\pi}} + \log(2)$$

and, as $\Gamma'(n) = \Gamma(n)\psi(n)$, where $\psi(n)$ is the *digamma* function

$$E\left[\log(\epsilon_n^2)\right] = \psi(1/2) + \log(2) \approx -\gamma - \log(2)$$

where γ is the *Euler-Mascheroni* constant.

Substituting this value in (3.3), it is obtained that

$$E\left[\log(\lambda\epsilon_n^2)\right] = \log(\lambda) + E\left[\log(\epsilon_n^2)\right] = \log(\lambda) - \gamma - \log(2) < 0$$

and, hence, $\lambda < 2e^{\gamma}$. $\square$

In this case, as estimations in data cloning methods are not quite affected by prior distributions (see Lele et al. (2007)), we consider vaguely informative uniform distributions, which include the corresponding restrictions of Proposition 1,

$$\begin{aligned} \beta &\sim U(0, d), \\ \lambda &\sim U(0, 2e^{\gamma}), \\ \delta &\sim U(0, 1), \end{aligned} \tag{3.4}$$

where d is any positive value.

The corresponding joint posterior distribution is the product of the previous prior distributions and the k^{th} power of the pseudo-likelihood function (see e.g. Francq and Zakoian (2010))

$$\begin{aligned} \pi^{(k)}(\beta, \lambda, \delta|\mathbf{y}) &\propto [L(\beta, \lambda, \delta|\mathbf{y})]^k \, \pi(\beta)\pi(\lambda)\pi(\delta) \\ &\propto \prod_{i=1}^{N} \left(\sqrt{2\pi\sigma_i^2}\right)^{-k} \exp\left\{-k\frac{\epsilon_i^2}{2\sigma_i^2}\right\} I_{(0,d)}(\beta) I_{(0,2e^{\gamma})}(\lambda) I_{(0,1)}(\delta) \end{aligned}$$

Then, the conditional posterior distribution of the parameters are

$$\begin{aligned} \pi^{(k)}(\beta|\lambda, \delta, \mathbf{y}) &\propto \left(\prod_{i=1}^{N} \left(\sqrt{2\pi\sigma_i^2}\right)^{-k}\right) \exp\left\{-k\frac{\epsilon_i^2}{2\sigma_i^2}\right\} I_{(0,d)}(\beta) \\ \pi^{(k)}(\lambda|\beta, \delta, \mathbf{y}) &\propto \left(\prod_{i=1}^{N} \left(\sqrt{2\pi\sigma_i^2}\right)^{-k}\right) \exp\left\{-k\frac{\epsilon_i^2}{2\sigma_i^2}\right\} I_{(0,2e^{\gamma})}(\lambda) \\ \pi^{(k)}(\delta|\beta, \lambda, \mathbf{y}) &\propto \left(\prod_{i=1}^{N} \left(\sqrt{2\pi\sigma_i^2}\right)^{-k}\right)) \exp\left\{-k\frac{\epsilon_i^2}{2\sigma_i^2}\right\} I_{(0,1)}(\delta) \end{aligned}$$

The data cloning algorithm starts with an initial solution $\beta^{(0)}$, $\lambda^{(0)}$, $\delta^{(0)}$, and generates values for $\beta^{(m)}$, $\lambda^{(m)}$, $\delta^{(m)}$ from the previous conditional posterior distributions of the parameters in each iteration m.

The algorithm can be summarized in the next steps

Step 1: Set initial values $\beta^{(0)}, \lambda^{(0)}, \delta^{(0)}$ at $m = 0$.

Step 2: Generate $\beta^{(m+1)}$ from

$$\beta^{(m+1)} \sim \pi^{(k)}(\beta|\lambda^{(m)}, \delta^{(m)}, \mathbf{y})$$

Step 3: Generate $\lambda^{(m+1)}$ from

$$\lambda^{(m+1)} \sim \pi^{(k)}(\lambda|\beta^{(m+1)}, \delta^{(m)}, \mathbf{y})$$

Step 4: Generate $\delta^{(m+1)}$ from

$$\delta^{(m+1)} \sim \pi^{(k)}(\delta|\beta^{(m+1)}, \lambda^{(m+1)}, \mathbf{y})$$

Step 5: Accept or reject the proposed values $\beta^{(m+1)}$, $\lambda^{(m+1)}$ and $\delta^{(m+1)}$ according to the Metropolis Hasting updates. Set $m = m + 1$ and go to Step 2.

The algorithm has been programmed using `Jags` (Plummer et al. (2003)) software in all cases by means of the package `runjags` (Denwood (2011)) from the `R` project (R Core Team (2012)). One advantage of using `Jags` is that it constructs the full conditional distributions and it carries out the Gibbs sampling from the model specifications. We have programmed the algorithm of data cloning by using package `dclone` (Sólymos (2010)) from the `R` project (R Core Team (2012)). All codes are available in appendix A.5.

3.3 Application of data cloning methodology for GARCH models

In this section, we undertake first a simulation study to check the estimator obtained by a data cloning methodology in a GARCH(1,1) model. Then, we compare the results with a MCMC standard methodology and, finally, we consider a real data application.

3.3.1 Simulation study of data cloning with simulated GARCH data

We simulate, as an example, 1000 observations from a GARCH(1,1) model with parameters $\beta = 3$, $\lambda = 0.7$ and $\delta = 0.1$, by means of the `R` library `TSA` (Chan and Ripley (2012)). Simulated data are plotted in figure 3.1.

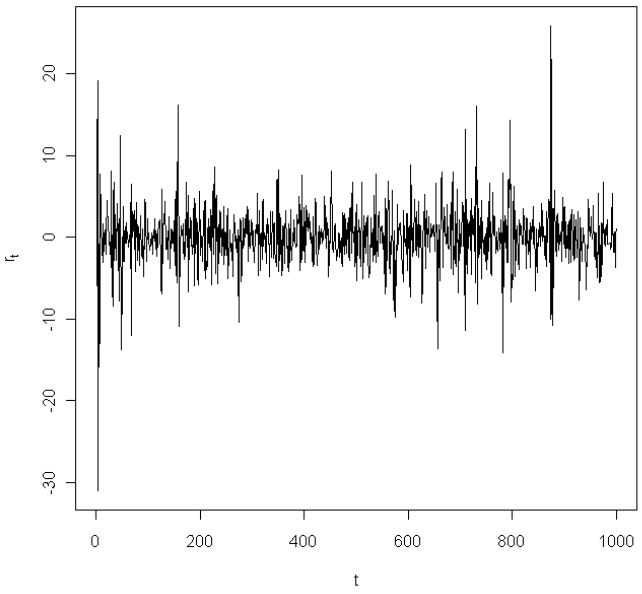

Figure 3.1: Simulated GARCH process

Before applying the data clone method, we check whether the parameters are estimable and we try to find out what is the optimal number of clones to be used. We use several diagnosis measures, that are included in the library `dclone`. Namely, the function `dcdiag` calculates some statistics, like the maximum eigenvalue of the posterior variance (Lele et al. (2010)) and the $\hat{R}$ statistics (Brooks and Gelman (1998)). Table 3.1 shows the obtained values for these statistics, when we apply the estimator with different number of clones over the simulated data.

number clones	λ_{max}	MSE	R^2	$\hat{R}$
10	0.011140698	0.0018738148	0.0003029090	1.001787
20	0.005360305	0.0047266814	0.0007291688	1.000445
30	0.003573252	0.0019818529	0.0003019833	1.000361
40	0.002787538	0.0010856731	0.0001592909	1.000453
50	0.002212842	0.0044508016	0.0005863694	1.000960
60	0.001813621	0.0038002256	0.0003086169	1.000251
70	0.001544969	0.0044780238	0.0007456815	1.000789
80	0.001351319	0.0166057846	0.0015524794	1.004203
90	0.001224819	0.0028387457	0.0002602986	1.001871
100	0.001072735	0.0009801083	0.0001240946	1.000311

Table 3.1: Diagnosis table for the simulated data

The λ_{max} value is based on the largest eigenvalue of the posterior variance-covariance matrix. Lele et al. (2010) proved that the maximum eigenvalue gives a hint of the degeneration of the posterior distribution, and if it is closer to zero, prior distributions have less influence over the results. This measure can also be used to check if the parameters of the model are estimable.

By the other hand, if the maximum eigenvalue converges to zero as the number of clones increases, then the parameters are estimable. In table 3.1 it may be observed that as the number of clones increases, the maximum eigenvalue of the posterior variance covariance matrix (λ_{max}) converges to zero. Then, it guarantees that the parameters of the model are estimable. Furthermore, for 10 clones the value of this statistic is small enough to assume that results will be accurate. Moreover, if the number of clones is larger than 20, the improvements in the estimations are not quite relevant.

Others useful measures related with the selection of the optimal number of clones, are the mean squared error (MSE) and the R^2 statistic. They are both based on a χ^2 approximation and should converge to zero when the number of clones increases. In table 3.1 it may be noted that MSE and R^2 converge to zero

as the number of clones increases. As in case of λ_{max}, this fact provides more evidences about the estimability of the parameters. They also show no relevant improvements in them, when assuming a number of clones larger than 10.

The last column of table 3.1 includes the $\hat{R}$ criteria, which was introduced by Brooks and Gelman (1998). The interpretation of this measure is similar to the previous ones; it converges to one as the number of clones increases. In this example, with more than 10 clones there are not significant improvements regarding the convergence to one.

As we previously mentioned, vaguely informative distributions have been used as prior distributions for all parameters, assuming that we have no relevant information about them. The prior distributions are shown in (3.4); here we have set $d = 100$.

Consequently, after setting the number of clones to 10 we apply the data cloning methodology to the simulated GARCH(1,1) data. Table 3.2 shows the real value of the parameters, the obtained estimations, their standard errors and the approximate frequentist confidence intervals for β, λ and δ.

Parameter	Real Value	Estimator	S.D.	95% Confidence Interval
β	3	3.229	0.333	(2.589, 3.869)
λ	0.7	0.744	0.075	(0.597, 0.891)
δ	0.1	0.088	0.031	(0.027, 0.148)

Table 3.2: Estimation of the GARCH parameters using the data cloning methodology

Table 3.2 shows that all the parameters are quite close to their real values. Furthermore, they have small standard errors and the confidence intervals include the true values of the parameters in all cases. Similar results can be found with any other set of possible values.

Now we compare the obtained data cloning estimators for the model GARCH(1,1), with the MCMC estimator for the same model introduced by Nakatsuma (1998). Table 3.3 shows the estimations for the same simulated series using a MCMC approach.

Parameter	Real values	Posterior means	S.D.	HPD 0.95
β	3	3.226	0.328	(2.612, 3.898)
λ	0.7	0.743	0.074	(0.606, 0.896)
δ	0.1	0.094	0.031	(0.042, 0.163)

Table 3.3: Estimation of the GARCH parameters using MCMC

Comparing table 3.2 and table 3.3 it may be observed that all estimators provide estimations which are really close to the real values. Consequently, we can consider that both methodologies produce accurate estimations, although standard errors are smaller for the data cloning estimates than for the MCMC ones.

3.3.2 Data cloning estimations for real data using a GARCH(1,1) model

In this section, we use a data cloning procedure to estimate the parameters of a GARCH(1,1) model over a real data set. We try to analyse the behaviour of the weekly returns of the NASDAQ 100 stock index from January 2000 to November 2012. The series is showed in figure 3.2.

The first step is being sure that this data set can be properly modelized by using a GARCH(1,1) model, so we start applying different techniques to ensure it. In this sense, it is necessary observe the autocorrelation function and the partial autocorrelation function, and they are shown in figures 3.3 and 3.4 respectively.

The autocorrelation function of the returns (3.3) shows that a strong white

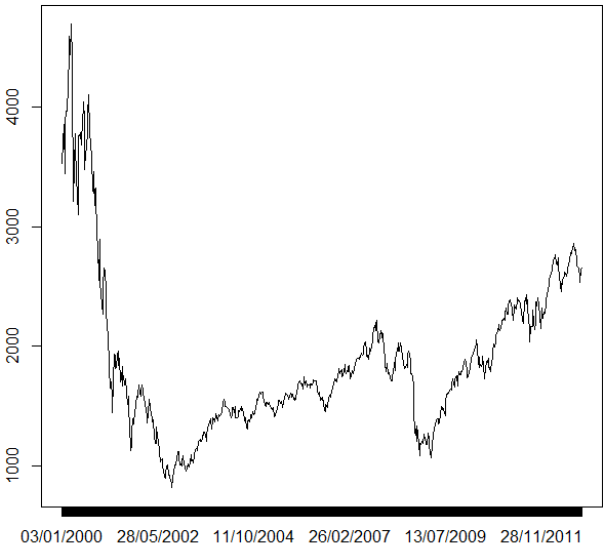

Figure 3.2: Weekly returns of Nasdaq 100 from January 2000 to November 2012

noise assumption cannot be sustained, because some of the sample autocorrelations are outside of the 95% significance bands. But these significance bands are not valid for the case of weak white noise (see Francq and Zakoian (2010)).

Figure 3.5 shows the autocorrelation function with the corresponding bands for weak white noise. It may be observed that, assuming these bands, the assumption about weak white noise is sustained for all lags except the first one, fact that is typical when assuming GARCH effects.

Now we apply the unitary roots test of Dickey and Fuller (1979). This test is used to detect the presence of unitary roots in time series, and consequently to ensure the stationarity of the series. It tests the null hypothesis about the

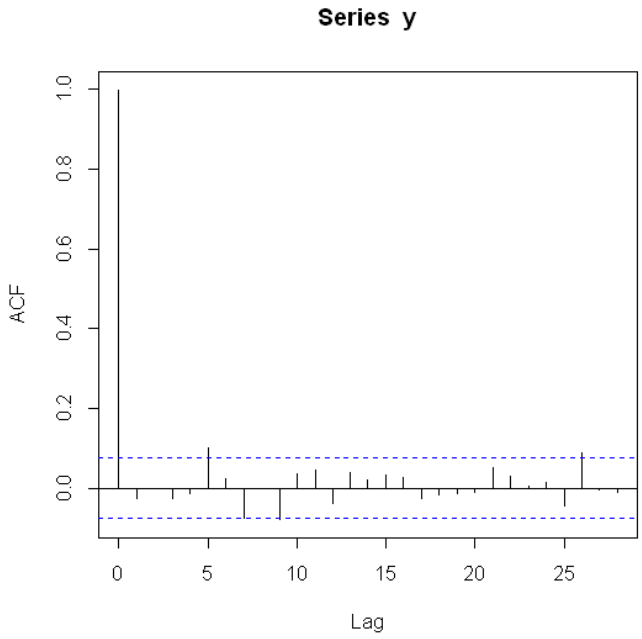

Figure 3.3: Autocorrelation Function for the weekly return of Nasdaq 100 from January 2000 to November 2012

absence of unitary roots in terms of

$$H_0 \quad : \quad \delta = 0$$

$$H_1 \quad : \quad \delta \neq 0$$

Where $\delta = \rho - 1$ being ρ the coefficient associated to the first lag in the model. This test is applied on residuals, and it is not possible use a standard t-distribution to obtain the critical values. Instead, it is used a constructed Dickey-Fuller table. If a time series presents unitary roots, then it may be deduced that it does not present stationarity.

The Dickey fuller statistic for the observed Nasdaq returns series is equal to -26.7817, therefore we can reject the null hypothesis of non stationarity at a significance level of 0.01 and we can conclude that the series has no unitary roots, and consequently it is stationary.

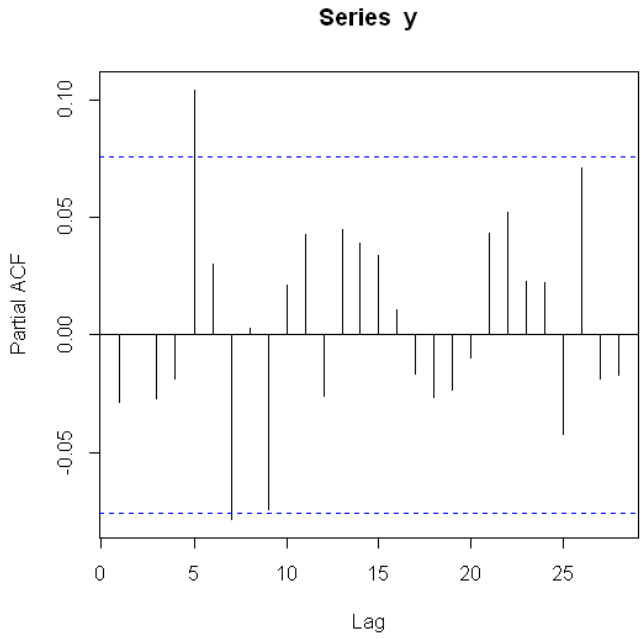

Figure 3.4: Partial Autocorrelation Function for the weekly return of Nasdaq 100 from January 2000 to November 2012

The next step is looking for autocorrelation evidences in the series. The most common test for checking if a series is a strong white noise or presents autocorrelation, is the Ljung and Box (1978) test, which evaluates the null hypothesis about absence of autocorrelation, using the following statistic

$$Q_m^{LB} := n(n+2)\sum_{i=1}^{m}\frac{\hat{\rho}^2}{n-i}.$$

Under the null hypothesis, the asymptotic distribution of Q_m^{LB} is a chi-square distribution χ^2_m. This test evaluates the absence of autocorrelation in the model. Then, if a series is a realization of strong white noise, the asymptotic distribution of the statistic is a chi-squared χ^2_m. Although, in the case of GARCH models the test has to be modified because it is not robust for conditional heteroscedasticity.

Table 3.4 shows the results of the Ljung-Box test in the Nasdaq returns series. In this series we have to reject the null hypothesis, although it does not mean

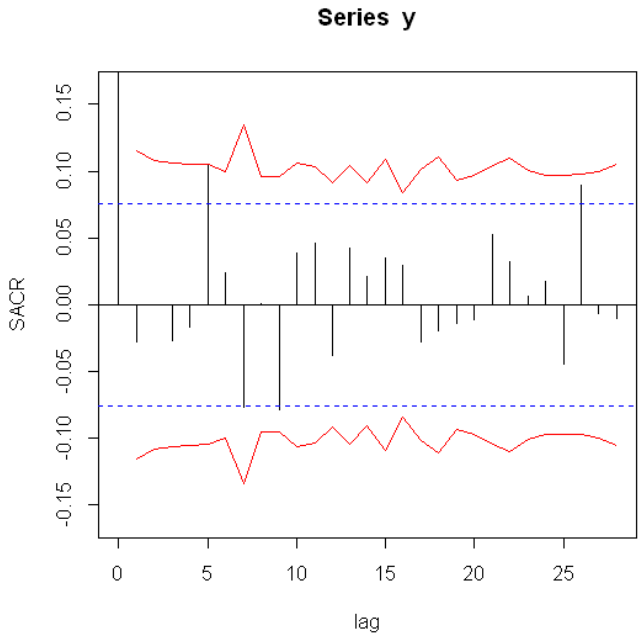

Figure 3.5: Autocorrelation Function for the weekly return of Nasdaq 100 from January 2000 to November 2012 with weak white noise bands

that the strong white noise assumption should not be rejected because, as we mention, this test is not robust in presence of conditional heteroscedasticity, and therefore it is useless in this situation.

We also use the Lagrange Multipliplier (LM) test introduced by Godfrey (1978) and Breusch (1978), in order to check the presence of autocorrelation in the series. Initially, this test was proposed for testing the null hypothesis of absence of autocorrelation against the alternative about the presence of AR(p) or MA(q) effects. The hypotheses are usually expressed as

$$\begin{aligned} H_0 &: \rho_1 = \rho_2 = \cdots = \rho_r = 0 \\ H_1 &: \exists i, \quad \rho_i \neq 0 \end{aligned}$$

Engle (1982) recommended to use this test also when the alternative hypothesis is the presence of ARCH effects. If we are are testing this presence of ARCH

lags	Q_m^{LB}	*df*	p-value
5	8.848119	5	0.11688312
10	18.864095	10	0.06393606
15	24.010714	15	0.07792208
20	25.798445	20	0.16883117
25	30.359635	25	0.21178821
30	37.570483	30	0.17482517

Table 3.4: Portmanteau test of Nasdaq returns

effects, we can use the same test, but assuming that the alternative hypothesis points out the presence of ARCH effects in the process.

The statistic of this contrast is $n \cdot R^2$, where n is the number of observations in the sample and R^2 is the determination coefficient of an auxiliary model which includes, as dependent variable, the residuals of the model and, as independent variables, those of the original model and the number of lags of the residuals.

When the the null hypothesis is true the statistic has a χ_r^2 distribution, where r is the number of lags included in the auxiliary model.

lags	*LM*	*df*	p-value
1	20.2287	1	6.871e-06
5	40.6069	5	1.126e-07
10	82.096	10	1.947e-13
15	179.5892	15	2.2e-16
20	212.5089	20	2.2e-16

Table 3.5: Lagrange Multiplier test for Nasdaq returns

Table 3.5 shows the *LM* statistic and the corresponding p-values computed for the Nasdaq returns. With all the selected number of lags, the null hypothesis of absence of autocorrelation has to be rejected, therefore we can conclude that this series presents ARCH effects.

After assuming that series may be modelled using an ARCH or GARCH models, we use the Akaike Information Criterion (AIC) (see Akaike (1974)) and the Bayesian Information Criterion (BIC) (see Schwarz (1978)) to select the best orders for the ARCH or GARCH model.

In order to find the best ARCH or GARCH model to describe the Nasdaq series, we compare 6 possible parsimonious models: ARCH(1), ARCH(2), GARCH(1,1), GARCH(1,2), GARCH(2,1) and GARCH(2,2), using the AIC and BIC criteria. The values for both measures for the different models are shown in table 3.6.

Model	AIC	BIC
ARCH(1)	2381.906	-2372.885
ARCH(2)	-2538.131	-2524.601
GARCH(1,1)	-2623.244	-2609.713
GARCH(1,2)	-2614.217	-2596.176
GARCH(2,1)	-2606.293	-2588.252
GARCH(2,2)	-2611.563	-2589.012

Table 3.6: Akaike information Criterion and Bayesian Information Criterion for several GARCH models

The minimum values for both AIC and BIC criteria are found for the GARCH(1,1) model, so we can conclude this is the best model to describe the considered Nasdaq series.

As a complementary approach we can test a z-statistic, to test the individual significance of the parameters. The values of z-statistics are computed as the ratio between the estimated parameters and their standard deviations. Here, the z-statistics and p-values of parameters β, λ and δ of the GARCH(1,1) model are shown in table 3.7. It may be noted that all estimates are significantly different of zero at a 0.01 level. By the other hand, if we consider other GARCH or ARCH models with a higher number of terms, the corresponding parameters are found

to be no significant.

Parameter	z-statistics	p-value
β	2.59	0.004772993
λ	4.64	1.691592e-06
δ	34.08	6.112542e-255

Table 3.7: Individual significance test for GARCH(1,1) parameters

Then, we will use the data cloning approach, as in section 3.3, to estimate the model parameters, their standard errors and the approximate maximum likelihood intervals. As before, we will consider as prior distribution the vaguely informative uniform distributions shown in (3.4), seting $d = 100$.

First, we check what is the optimal number of clones (see Sólymos (2010)) and we obtain that there are not relevant improvements when the number of clones is larger than 40; therefore we use this number of clones to analyse the series.

Using 40 clones, the results are shown in Table 3.8.

Parameter	Estimated Value	S.D.	Confidence Interval 95%
β	$7.82e$-05	$2.97e$-06	($4.13e$-05 ; $1.15e$-04)
λ	0.196	$5.01e$-03	$(0.134; 0.258)$
δ	0.741	$4.60e$-03	$(0.684; 0.798)$

Table 3.8: Clone estimations for GARCH(1,1) parameters to model Nasdaq weekly returns

Results obtained with the data cloning approach can be compared with those obtained by a standard MCMC Bayesian approach (see e.g. Nakatsuma (1998)). Table 3.9 shows the posterior means, standard errors and the credible intervals for the GARCH(1,1) parameters under a full Bayesian approach.

Consequently, as with simulated series, both data cloning and Bayesian methodologies show similar results (see table 3.8 and table 3.9).

Parameter	Estimated Value	S.D.	HPD 0.95
β	$8.31e$-05	$1.94e$-05	($4.90e$-05 ; $1.25e$-04)
λ	0.2054	0.0324	$(0.147; 0.2748)$
δ	0.7324	0.0285	$(0.675; 0.7863)$

Table 3.9: MCMC estimations for GARCH(1,1) parameters to model Nasdaq weekly returns

In GARCH processes, $(\lambda + \delta)$ measures the persistence of the volatility of the process, which is the tendency for moving in the same direction. When this sum is close to 1, it can be said that the volatility is *persistent*. The estimated values for these parameters, by means of data cloning or Bayesian methods, have in both cases a sum near to 1, so we can conclude that the volatility in the Nasdaq weekly returns between 2000 and 2012 is *persistent*.

3.4 Data cloning estimator for COGARCH Model parameters

In this section we introduce data cloning methodology to estimate the parameters of the COGARCH model described in section 1.4. In order to do it, we consider the approach of Maller et al. (2008) described in 2.2.

We set similar vaguely informative prior distributions, as we considered in section 2.3

$$\begin{aligned} \beta &\sim U(0, a) \\ \varphi &\sim U(0, c) \\ \eta &\sim U(\varphi, b) \end{aligned} \tag{3.5}$$

where a, c and b are positive real values and $b > c$.

The corresponding joint posterior distribution is the product of the previous

prior distributions and the k$^t h$ power of the pseudo-likelihood function (see 2.9) given by

$$\begin{aligned}\pi^{(k)}(\beta,\varphi,\eta|\mathbf{y}) &\propto [L(\beta,\varphi,\eta|\mathbf{y})]^k\,\pi(\beta)\pi(\varphi)\pi(\eta)\\ &\propto \left(\prod_{i=1}^{N}\rho_i^2\right)^{-k/2}\exp\left\{-k\sum_{i=1}^{N}\frac{y_i^2}{2\rho_i^2}\right\}I_{(0,a)}(\beta)I_{(\varphi,b)}(\eta)I_{(0,c)}(\varphi).\end{aligned}$$

Then, the conditional posterior distributions of parameters are

$$\begin{aligned}\pi^{(k)}(\beta|\varphi,\eta,\mathbf{y}) &\propto \left(\prod_{i=1}^{N}\rho_i^2\right)^{-k/2}\exp\left\{-k\sum_{i=1}^{N}\frac{y_i^2}{2\rho_i^2}\right\}I_{(0,a)}(\beta)\\ \pi^{(k)}(\varphi|\beta,\eta,\mathbf{y}) &\propto \left(\prod_{i=1}^{N}\rho_i^2\right)^{-k/2}\exp\left\{-k\sum_{i=1}^{N}\frac{y_i^2}{2\rho_i^2}\right\}I_{(0,c)}(\varphi)\\ \pi^{(k)}(\eta|\beta,\varphi,\mathbf{y}) &\propto \left(\prod_{i=1}^{N}\rho_i^2\right)^{-k/2}\exp\left\{-k\sum_{i=1}^{N}\frac{y_i^2}{2\rho_i^2}\right\}I_{(\varphi,b)}(\eta)\end{aligned}$$

Data cloning algorithm generates values for $\beta^{(m)}$, $\varphi^{(m)}$, $\eta^{(m)}$ from the previous conditional posterior distributions of the parameters in each iteration m from initial values $\beta^{(0)}$, $\varphi^{(0)}$, $\eta^{(0)}$.

The algorithm can be summarized in the next steps

Step 1: Set initial values $\beta^{(0)},\varphi^{(0)},\eta^{(0)}$ at $m=0$.

Step 2: Generate $\beta^{(m+1)}$ from

$$\beta^{(m+1)} \propto \left(\prod_{i=1}^{N}\rho_i^2\right)^{-k/2}\exp\left\{-k\sum_{i=1}^{N}\frac{y_i^2}{2\rho_i^2}\right\}I_{(0,a)}(\beta)$$

where

$$\begin{aligned}\rho_i^2 &= \left(\sigma_{i-1}^2-\frac{\beta^{(m)}}{\eta^{(m)}-\varphi^{(m)}}\right)\left(\frac{\exp\{(\eta^{(m)}-\varphi^{(m)})\Delta t_i\}-1}{\eta^{(m)}-\varphi^{(m)}}\right)+\frac{\beta^{(m)}\Delta t_i}{\eta^{(m)}-\varphi^{(m)}}\\ \sigma_{i-1}^2 &= \beta^{(m)}\Delta t_{i-1}+\exp\{-\eta^{(m)}\Delta t_{i-1}\}\sigma_{i-2}^2+\varphi^{(m)}\exp\{-\eta^{(m)}\Delta t_{i-1}\}y_{i-1}^2\\ \sigma_0^2 &= \frac{\beta^{(m)}}{\eta^{(m)}-\varphi^{(m)}}\end{aligned}$$

for $i=1,\ldots,N$.

Step 3: Generate $\varphi^{(m+1)}$ from

$$\varphi^{(m+1)} \propto \left(\prod_{i=1}^{N} \rho_i^2\right)^{-k/2} \exp\left\{-k \sum_{i=1}^{N} \frac{y_i^2}{2{\rho_i}^2}\right\} I_{(0,c)}(\varphi)$$

where

$$\begin{aligned}
\rho_i^2 &= \left(\sigma_{i-1}^2 - \frac{\beta^{(m+1)}}{\eta^{(m)} - \varphi^{(m)}}\right)\left(\frac{\exp\{(\eta^{(m)} - \varphi^{(m)})\Delta t_i\} - 1}{\eta^{(m)} - \varphi^{(m)}}\right) \\
&\quad + \frac{\beta^{(m+1)}\Delta t_i}{\eta^{(m)} - \varphi^{(m)}} \\
\sigma_{i-1}^2 &= \beta^{(m+1)}\Delta t_{i-1} + \exp\{-\eta^{(m)}\Delta t_{i-1}\}\sigma_{i-2}^2 \\
&\quad + \varphi^{(m)} \exp\{-\eta^{(m)}\Delta t_{i-1}\} y_{i-1}^2 \\
\sigma_0^2 &= \frac{\beta^{(m+1)}}{\eta^{(m)} - \varphi^{(m)}}
\end{aligned}$$

for $i = 1, \ldots, N$.

Step 4: Generate $\eta^{(m+1)}$ from

$$\eta^{(m+1)} \propto \left(\prod_{i=1}^{N} \rho_i^2\right)^{-k/2} \exp\left\{-k \sum_{i=1}^{N} \frac{y_i^2}{2{\rho_i}^2}\right\} I_{(\varphi^{(m+1)},b)}(\eta)$$

where

$$\begin{aligned}
\rho_i^2 &= \left(\sigma_{i-1}^2 - \frac{\beta^{(m+1)}}{\eta^{(m)} - \varphi^{(m+1)}}\right)\left(\frac{\exp\{(\eta^{(m)} - \varphi^{(m+1)})\Delta t_i\} - 1}{\eta^{(m)} - \varphi^{(m+1)}}\right) \\
&\quad + \frac{\beta^{(m+1)}\Delta t_i}{\eta^{(m)} - \varphi^{(m+1)}} \\
\sigma_{i-1}^2 &= \beta^{(m+1)}\Delta t_{i-1} + \exp\{-\eta^{(m)}\Delta t_{i-1}\}\sigma_{i-2}^2 \\
&\quad + \varphi^{(m+1)} \exp\{-\eta^{(m)}\Delta t_{i-1}\} y_{i-1}^2 \\
\sigma_0^2 &= \frac{\beta^{(m+1)}}{\eta^{(m)} - \varphi^{(m+1)}}
\end{aligned}$$

for $i = 1, \ldots, N$.

Step 5: Accept or reject the proposed values $\beta^{(m+1)}$, $\eta^{(m+1)}$ and $\varphi^{(m+1)}$ according to the Metropolis Hasting updates. Set $m = m + 1$ and go to Step 2.

The algorithm to estimate the COGARCH(1,1) parameters has been programmed using the `Jags` (Plummer et al. (2003)) software by means of the package `runjags` (Denwood (2011)) from the `R` project (R Core Team (2012)). Additionally, we have also used the package `dclone` (Sólymos (2010)) from the `R` project (R Core Team (2012)) to program the data cloning algorithm. Codes are available in appendix A.6.

3.5 Applications of the data cloning methodology for COGARCH models

In this section, we undertake first a simulation study to check the estimators obtained by using data cloning methodology in a COGARCH(1,1) model. Then, we compare the results with the MCMC standard methodology introduced in section (2.3) and, finally, we consider a real data application and compare its results with the MCMC standard estimates.

3.5.1 Simulation study of data cloning with simulated COGARCH data

In this section, we undertake a simulation study to check the estimators obtained by a data cloning methodology in a COGARCH(1,1) model. We are going to use again the simulated sample path we used with our MCMC estimator in order to be able to compared the obtained results with the results using Data Cloning methodology. It is a sample path of COGARCH process with parameters $\beta = 0.25$, $\eta = 0.35$ and $\varphi = 0.02$ simulated by using the code in `R` in Appendix A.2. The sample path is showed in figure (2.1). Figures (2.2) and (2.3) showed the returns and the volatility respectively.

To apply the Data cloning algorithm to estimate COGARCH paramaters we used the uninformative prior distributions defined in (3.5). We set $a = 100000$ and $b = 100000$ to be large enough. As we remark in Chapters 1 and 2, bigger values for φ provide sample paths with extremely big jumps to be real data, so we set $c = 0,05$, which lets big enough jumps providing good estimates.

Similarly as in the case of GARCH estimation, before applying the data clone procedure, we analyse the diagnosis measures to check if parameters are estimable and to find what is the optimal number of clones to implement the procedure. Table 3.10 shows a list of these statistics for several possible number of clones. Namely, the maximum eigenvalue of the posterior variance, the minimum squared error, the squared error and $\hat{R}$. As in section 3.3, we check in this way the convergence of the parameters and we fix the optimal number of clones to use.

number clones	λ_{max}	MSE	R^2	$\hat{R}$
2	0.026820930	0.8502252	0.08165483	1.017508
5	0.013555492	0.9026417	0.06761531	1.010057
10	0.007693267	0.6495554	0.04552857	1.005517
15	0.0044663527	1.0163490	0.06758207	1.005078
20	0.003534333	0.5404925	0.03392750	1.017282
25	0.002880896	0.6100857	0.04106375	1.030992
30	0.002500749	1.9218046	0.11143148	1.039526

Table 3.10: Diagnosis table of the data cloning estimator of a COGARCH(1,1) model

Here, the maximum eigenvalue of the posterior variance covariance matrix converges to zero when the number of clones increases, then, we can conclude that the parameters of the model are estimable (see table (3.10)). According to the mean squared error and the R^2 statistic, the optimal number of clones to be used is 20. Similar conclusion is observed by the $\hat{R}$ criterion, so we set the number of clones in the procedure as 20.

Parameter	Real Value	Estimated Value	S.D.	95% Confidence Interval
β	0.25	0.26795	0.21116	(0, 0.682)
η	0.35	0.18782	0.14597	(0, 0.474)
φ	0.02	0.04481	0.02122	(0.003, 0.086)

Table 3.11: Estimation of the COGARCH(1,1) parameters using the data cloning methodology

After applying the data cloning procedure, in table 3.11 it is shown the obtained estimations, their standard errors and the approximate confidence intervals. All estimations are closed to the real values of the parameters ($\beta = 0.35$, $\eta = 0.25$ and $\phi = 0.02$) and present small standard errors. Moreover, the confidence intervals include all real values of the parameters.

Parameter	Real Value	Posterior mean	S.D.	HPD 0.95
β	0.25	0.427	0.158	(0.138 ; 0.747)
η	0.35	0.301	0.132	(0.092 ; 0.600)
φ	0.02	0.029	0.014	(0.002 ; 0.049)

Table 3.12: Estimation of the COGARCH(1,1) parameters using MCMC

Now, we compare the obtained data cloning estimators for the model COGARCH(1,1), with the MCMC estimator for the same model described in section 2.3, by using the Maller et al. (2008) approach. Table 3.12 shows the posterior means, the standard deviations and the HPD intervals of the parameters obtained with standard MCMC algorithm applied to the same sample path.

When comparing table 3.11 with table 3.12, it may be noted that with both methodologies, accurate estimations of the parameters of the model are obtained and both, approximate frequentist confidence intervals and credible intervals, contain the real values of the parameters. The goal of Data Cloning methodology in this case is providing maximum likelihood estimations.

3.5.2 Data cloning estimations for real data using a COGARCH(1,1) model

In this section, we use a data cloning procedure to estimate the parameters of COGARCH(1,1) model over a real data. We consider the behaviour of the daily returns of Nasdaq index from January 2008 to December 2012. Here, data are not equally spaced because we consider weekends and holiday periods of time, and the approach of Maller et al. (2008) seams to be a natural procedure to model daily series. Figure (3.6) shows the evolution of these daily returns.

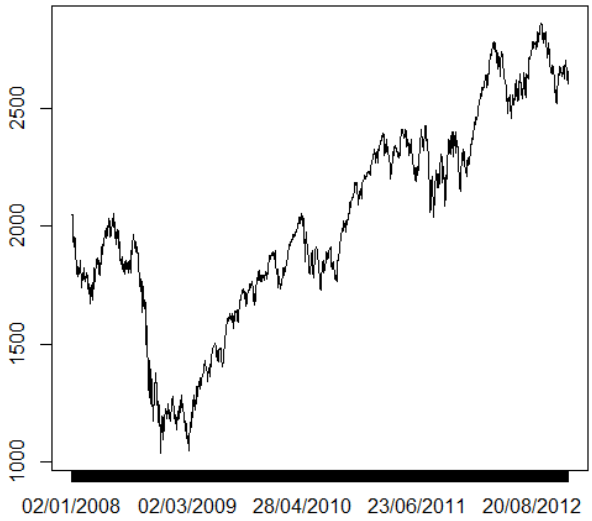

Figure 3.6: Daily evolution of Nasdaq 100 from January 2008 to November 2012

Then, we use the data cloning approach, as in section 3.5.1, to estimate the parameters, their standard errors and the approximate maximum likelihood intervals. First, we check what is the optimal number of clones (see (Sólymos (2010)) and we obtain that there are not relevant improvements when the number of clones is larger than 20 (see table (3.13)); therefore we use this number of clones

to analyse the series. The maximum eigenvalue showed in table (3.13) converges to zero with the increase of the number of clones, so it can be conclude that parameters are estimable.

number clones	λ_{max}	MSE	R^2	$\hat{R}$
5	0.169687	0.293721	0.027368	1.01772
10	0.070733	0.545003	0.051611	1.001016
15	0.036846	0.624237	0.050637	1.014734
20	0.024687	0.332539	0.028714	1.000976
25	0.018297	0.822261	0.059338	1.002529
30	0.012855	0.955443	0.067347	1.007905
35	0.010264	1.127749	0.082668	1.0079

Table 3.13: Diagnosis table of the data cloning estimator of a COGARCH(1,1) model over real data

Using 20 clones, the results are shown in Table 3.14, it shows the estimated values for the parameters, the standard deviation of the estimators and the approximate frequentist confidence intervals. As in the estimation for simulated data, the prior distributions we used are the weakly-informative uniform distributions defined in (3.5) setting $a = 100000$, $b = 100000$ and $c = 0,05$.

Parameter	Estimated Value	S.D.	Confidence Interval 95%
β	9.041	0.289	$(5.457; 12.626)$
η	0.055	0.00042	$(0.050; 0.061)$
φ	0.05	0.00015	$(0.048; 0.052)$

Table 3.14: Clone estimations of the COGARCH parameters to model the Nasdaq daily returns

Results obtained with the data cloning approach can be compared with those obtained by a standard MCMC Bayesian approach. Table 3.15 shows the posterior means, standard errors and the credible intervals for the COGARCH(1,1) parameters under a full Bayesian approach. Results with respect to Table 3.14 are similar, although standard errors seem to be smaller in the case of the data cloning approach.

Parameter	Estimated Value	S.D.	HPD 0.95
β	9.661	2.113	$(6.140; 14.414)$
η	0.054	0.0043	$(0.045; 0.063)$
φ	0.047	0.0028	$(0.039; 0.050)$

Table 3.15: MCMC estimations of COGARCH parameters to model the Nasdaq daily returns

Parameter η measures the speed of the decline of a volatility burst. For this dataset the η value is not very large, namely, when a volatility burst appears due to the arrival of new information to markets, its influence in volatility declines with a moderate speed.

Parameter ϕ measures the magnitude of the volatility jumps and it may be considered as a measure of how information affects to volatility and how fast market assumes new events. In Nasdaq-100 series the estimation of ϕ is quite large in relation to the values that it may reach in order to maintain stationarity. A possible interpretation is that volatility bursts are quite big and daily index moves largely up or down, regarding the appearance of relevant information in market.

Finally, parameter β represents the level of volatility and regarding the Nasdaq-100 values, the estimated value is not very large.

After using GARCH and COGARCH models to analyse the behaviour of the Nasdaq-100 index series, by using both data cloning and a MCMC Bayesian methodologies, we can conclude that the obtained results in this chapter can be easily interpreted from a practical point of view and they may be used for forecasting issues of financial series.

Chapter 4

Hybrid Monte Carlo Estimation for GARCH and COGARCH model parameters

4.1 Introduction to Hybrid Monte Carlo

The Hamiltonian dynamics methodology was introduced by Alder and Wainwright (1959) as an approach to molecular simulation which resulted to be asymptotically equivalent to MCMC methods. Both methodologies co-existed in the following decades despite they overlapped in their application areas. Eventually Duane et al. (1987) joined these two methodologies in what they called *Hybrid Monte Carlo* (HMC) methods, because it involves a combination of a MCMC approach with a deterministic step.

Originally, Hamiltonian dynamics was proposed to model the behaviour of a system of idealized molecules, then Duane et al. (1987) introduced HMC with the same purpose, although there were found new applications in Statistics, like Neal (1995) regarding neural networks, Ishwaran (1999) in generalized linear models and Schmidt (2009) in non-linear regression.

Due to their random walk behaviour, Gibbs sampler and Metropolis algorithms may be inefficient in some situations, like simulations can take a long time moving around the target distribution in complex models. HMC methods try to amend these drawbacks. In this chapter we introduce the inference in GARCH and COGARCH models based on the Hybrid Monte Carlo algorithm, which results to have a more efficient behaviour than the MCMC approach.

We first describe in the next section, the Hamiltonian dynamics and the Hybrid Monte Carlo algorithms. In the following sections, HMC estimators for GARCH and COGARCH parameters are introduced and their performance analysed.

4.2 Description of Hybrid Monte Carlo

Hamiltonian dynamics was initially applied in the Physics framework in order to model the behaviour of a *puck* moving on a surface. In this context two vectors are required: the position of the puck q and its velocity p. Both, position and velocity, depend on time. The potential energy function is given by $U(q)$ which is proportional to the height of the surface at the current position of the puck. The kinetic energy function is given by $K(q) = |p|^2 / (2m)$, being m the mass of the puck. It is well known that these two energies are related: when kinetic energy increases (decreases), potential energy decreases (increases). The total energy of the system is described by means of a function called *Hamiltonian function* $H(q, p)$.

Hamilton's equations are the partial derivatives of Hamiltonian function $H(q, p)$, and they describe the change in q and p over time:

$$\frac{dq_i(t)}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i(t)}{dt} = \frac{\partial H}{\partial q_i}$$

where $i = 1, \ldots, d$.

These equations are defined to map the movement of the puck in any time interval and in general they must be solved by numerical methods like the Euler procedure (see for a review of the main concepts Neal (2011)).

If these ideas are applied in the context of simulation, the position of the puck can be seen as the parameter of interest, i.e. $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$, and the potential energy will be the log-posterior probability density function of the variable of interest. For each component of θ_j there is an auxiliary *momentum* variable ϕ_j, in such a way that both are jointly updated for $j = 1, \ldots, d$.

The posterior density of $\boldsymbol{\theta}$ is first computed up to a multiplicative constant, then it is augmented by the distribution of the auxiliary parameter $\boldsymbol{\phi}$ in order to obtain the joint posterior distribution $P(\boldsymbol{\theta}, \boldsymbol{\phi}|y)$.

In the procedure, simulations are obtained from the joint posterior distribution, although the main interest is focused in the simulated values of $\boldsymbol{\theta}$. Values of $\boldsymbol{\phi}$ correspond to an auxiliary variable introduced to enable the algorithm to move faster in the parameter space.

It is required to compute the posterior density and the gradient of the log-posterior density, which in practice must be computed by means of numerical differentiation. If $\boldsymbol{\theta}$ is a d-dimensional parameter the gradient is defined as,

$$\frac{d \log P(\theta/y)}{d\theta} = \left(\frac{d \log \pi(\theta) L(\theta/y)}{d\theta_1}, \ldots, \frac{d \log \pi(\theta) L(\theta/y)}{d\theta_d} \right)$$

In general, this vector is easy to be computed in most models and the momentum distribution can be chosen by the researcher.

The HMC algorithm can be used to simulate from the posterior joint distribution $P(\boldsymbol{\theta}, \boldsymbol{\phi}|y)$, which must be a continuous distributions on $\mathbb{R}^d$ and the partial derivatives of the logarithm of the density function must exist and they can be computed.

It is usual to take as the prior distribution of $\boldsymbol{\phi}$ a d-dimensional multivariate normal distribution where for each component $j = 1, \ldots, d$, $\phi_j \sim N(0, M_j)$ and the covariance matrix M_j is a diagonal matrix.

The HMC algorithm can be summarised in three steps. The first one addresses the updating of the momentum variable, in the second one it is updated the position and momentum variables, and the third one new values are accepted or rejected according to a Metropolis-Hasting algorithm. These steps do not change

the joint distribution of $(\boldsymbol{\theta}, \boldsymbol{\phi})$.

Step 1: New values of the momentum variable $\boldsymbol{\phi}$ are simulated from the prior distribution, a d-dimensional normal distribution.

Step 2: In the second step, a pair of values of $(\boldsymbol{\theta}, \boldsymbol{\phi})$ are simulated by using the *leapfrog* method scaled by a factor ϵ. Then, a vector $(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$ is proposed and it will be accepted or rejected in the next step according to a Metropolis-Hasting algorithm. The leapfrog method begins taking a half step to make a half-update of $\boldsymbol{\phi}$, then it alternates with $N-1$ full-updates of parameter $\boldsymbol{\theta}$, and it finishes with a half-update of parameter $\boldsymbol{\phi}$. It is named as *leapfrog* because it splits the momentum updates into half steps and it turns to be an approximation to the physical Hamiltonian dynamics, in which the position and momentum are defined in continuous time. The leapfrog method involves the following steps:

(a) Half-update $\boldsymbol{\phi}$, using the gradient of the log-posterior density

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \frac{1}{2}\epsilon \frac{d \log p\left(\boldsymbol{\theta}|y\right)}{d\boldsymbol{\theta}}$$

(b) Full update the position $\boldsymbol{\theta}$ given the value of the momentum variable $\boldsymbol{\phi}$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon M^{-1}\boldsymbol{\phi}$$

being M the covariance matrix of momentum distribution $P(\boldsymbol{\phi})$. Instead of avoid the use of ϵ and include it into M, which could be considered easier, we keep both just to make M constant and alter only ϵ when using the algorithm.

(c) Half-update of ϕ using again the gradient of the log-posterior density.

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \frac{1}{2}\epsilon \frac{d \log p\left(\boldsymbol{\theta}|y\right)}{d\boldsymbol{\theta}}$$

When ϵ is close to zero, the leapfrog algorithm keeps the joint density $P(\boldsymbol{\theta}, \boldsymbol{\phi}|y)$ invariant. Assuming that ϵ is finite, the joint density $P(\boldsymbol{\theta}, \boldsymbol{\phi}|y)$ is not constant during the leapfrog steps, but it changes slightly if ϵ is small enough.

Step 3: In the third step it is addressed an acception-rejection procedure based on a Metropolis-Hastings approach. We denote the values of the parameter and momentum values before a leapfrog step as $\left(\boldsymbol{\theta}^{m-1}, \boldsymbol{\phi}^{m-1}\right)$ and $(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$ after it. Then, we calculate the ratio

$$r = \frac{p(\boldsymbol{\theta}^*|y)\, p(\boldsymbol{\phi}^*)}{p\left(\boldsymbol{\theta}^{m-1}|y\right) p\left(\boldsymbol{\phi}^{m-1}\right)},$$

in order to accept or reject the new values according to a Metropolis-Hasting rule

$$\boldsymbol{\theta}^m = \begin{cases} \boldsymbol{\theta}^* & \text{with probability } \min(r, 1) \\ \boldsymbol{\theta}^{m-1} & \text{Otherwise} \end{cases}$$

In strict sense, we should also set a new value to ϕ^m, but as it is an auxiliary variable it is not necessary.

This algorithm should be repeated until convergence that can be assessed based on a ANOVA criterion (see e.g. Gelman et al. (2014)).

The Hybrid Monte Carlo method is designed to work with target densities with positive values. If in any iteration, the algorithm takes a value of zero it is necessary to stop and start a new iteration considering the previous value of θ. If it takes negative values, the sign of ϕ must change in order to obtain positive values, then the algorithm preserves the balance and it is more efficient than if these values were just rejected. By the other hand, any change and movements to different values of $(\boldsymbol{\theta}, \boldsymbol{\phi})$ are possible. The algorithm can be tuned considering three different measures:

(*i*) The probability distribution of the momentum variable $\boldsymbol{\phi}$, usually by default it is assumed a normal distribution with zero mean and a given variance-covariance matrix.

(*ii*) The value of the factor ϵ for the leapfrog steps.

(*iii*) The number N of leapfrog steps per iteration. In order to calibrate the algorithm to the target distribution, it is necessary to meet the condition $\epsilon \cdot N = 1$ (e.g. $\epsilon = 0.1$ and $N = 10$).

These tuning parameters can be set previously or they may be changed at random for avoiding the algorithm to get stuck, taking into account information from the previous iterations. In general, it is convenient to take a previous warm-up period to tune the parameters, after that values from this period are discarded for the estimation of parameters. By the other hand, in practice, it is suitable to obtain an acceptance rate of 65%.

4.3 Hybrid Monte Carlo estimator for GARCH model parameters

In this section we introduce the Hybrid Monte Carlo methodology applied for the GARCH(1,1) model as defined in section 1.3.

The parameters of interest are $\boldsymbol{\theta} = (\beta, \lambda, \delta)$, which play the role of variables of position when describing the HMC algorithm.

The potential energy is the logarithm of the posterior density function

$$\frac{d \log P(\beta, \lambda, \delta|Y)}{d(\beta, \lambda, \delta)} = \left(\frac{d \log P(\beta, \lambda, \delta|Y)}{d\beta}, \frac{d \log P(\beta, \lambda, \delta|Y)}{d\lambda}, \frac{d \log P(\beta, \lambda, \delta|Y)}{d\delta} \right) \tag{4.1}$$

We consider vaguely informative prior distributions as in section 3.2 which include the corresponding restrictions of the parameters, namely,

$$\beta \sim U(0, d),$$
$$\lambda \sim U(0, 2e^{\gamma}),$$
$$\delta \sim U(0, 1),$$

where d is any positive value, and γ is the Euler-Mascheroni constant.

Then, the joint posterior distribution is

$$\pi(\beta, \lambda, \delta|Y) \propto \prod_{n=1}^{N} \left(\sqrt{2\pi\sigma_n^2}\right)^{-1} \exp\left\{-\frac{y_n^2}{2\sigma_n^2}\right\} \cdot I_{(0,d)}(\beta) \cdot I_{(0,2e^{\gamma})}(\lambda) \cdot I_{(0,1)}(\delta)$$

where $\sigma_n^2 = \beta + \lambda Y_{n-1}^2 + \delta\sigma_{n-1}^2$.

The momentum variables (ϕ_1, ϕ_2, ϕ_3) are auxiliary variables which follow a standard multivariate normal distribution,

$$\phi = (\phi_1, \phi_2, \phi_3) \sim N(0, I).$$

In order to calculate the corresponding potential energy (4.1), we define an intermediate function

$$g(\beta, \lambda, \delta) = \sum_{n=1}^{N} \left(\frac{1}{2}\log\left(2\pi\sigma_n^2\right) - \frac{y_n^2}{2\sigma_n^2}\right) = \frac{N}{2}\log(2\pi) + \frac{1}{2}\sum_{n=1}^{N}\left(\log\left(\sigma_n^2\right) - \frac{y_n^2}{2\sigma_n^2}\right) =$$

$$\frac{N}{2}\log(2\pi) + \frac{1}{2}\sum_{n=1}^{N}\left(\log\left(\beta + \lambda y_{t-1}^2 + \delta\sigma_{n-1}^2\right) - \frac{1}{2}y_n^2\left(\beta + \lambda y_{n-1}^2 + \delta\sigma_{n-1}^2\right)^{-1}\right)$$

The derivatives for $t = 1$ is

$$\begin{aligned}
\frac{dg(\beta, \lambda, \delta)}{d\beta} &= \frac{2\delta\sigma_0^2 + 2\lambda y_0^2 + y_n^2 + 2\beta}{2\left(\delta\sigma_0^2 + \lambda y_0^2 + \beta\right)^2} \\
\frac{dg(\beta, \lambda, \delta)}{d\lambda} &= \frac{1}{2}\frac{y_0^2\left(2\delta\sigma_0^2 + 2\lambda y_0^2 + y_n^2 + 2\beta\right)}{\left(\delta\sigma_0^2 + \lambda y_0^2 + \beta\right)^2} \\
\frac{dg(\beta, \lambda, \delta)}{d\delta} &= \frac{1}{2}\frac{\sigma_0^2\left(2\delta\sigma_0^2 + 2\lambda y_0^2 + y_n^2 + 2\beta\right)}{\left(\delta\sigma_0^2 + \lambda y_0^2 + \beta\right)^2}
\end{aligned}$$

For $n > 1$, the derivatives can be obtained by successive substitutions in σ_n^2, although cumbersome polynomial expressions are obtained which should be better managed by numerical methods.

Let us denote the corresponding partial derivatives of $g(\beta, \lambda, \delta)$ with respect to (β, λ, δ) as

$$\begin{aligned} D_\beta(\beta, \lambda, \delta) &= \frac{dg(\beta, \lambda, \delta)}{d\beta} \\ D_\lambda(\beta, \lambda, \delta) &= \frac{dg(\beta, \lambda, \delta)}{d\lambda} \\ D_\delta(\beta, \lambda, \delta) &= \frac{dg(\beta, \lambda, \delta)}{d\delta} \end{aligned}$$

In this way, the HMC algorithm is summarized as:

Step 1: Set initial values $(\beta^{(0)}, \lambda^{(0)}, \delta^{(0)})$ and $m = 0$.

Generate $(\phi_1^{(m)}, \phi_2^{(m)}, \phi_3^{(m)})$ from a $N(0, I)$.

Step 2: Generate proposed values for $(\beta^{(m+1)}, \lambda^{(m+1)}, \delta^{(m+1)}, \phi_1^{(m+1)}, \phi_2^{(m+1)}, \phi_3^{(m+1)})$ according to the leapfrog method, as follows

(a) $\phi_1^{(m+1/2)} \leftarrow \phi_1^{(m)} + \frac{1}{2}\epsilon D_\beta\left(\beta^{(m)}, \lambda^{(m)}, \delta^{(m)}\right)$

$\phi_2^{(m+1/2)} \leftarrow \phi_2^{(m)} + \frac{1}{2}\epsilon D_\lambda\left(\beta^{(m)}, \lambda^{(m)}, \delta^{(m)}\right)$

$\phi_3^{(m+1/2)} \leftarrow \phi_3^{(m)} + \frac{1}{2}\epsilon D_\delta\left(\beta^{(m)}, \lambda^{(m)}, \delta^{(m)}\right)$

(b) $\beta^{(m+1)} \leftarrow \beta^{(m)} + \epsilon M^{-1}\phi_1^{(m+1/2)}$

$\lambda^{(m+1)} \leftarrow \lambda^{(m)} + \epsilon M^{-1}\phi_2^{(m+1/2)}$

$\delta^{(m+1)} \leftarrow \delta^{(m)} + \epsilon M^{-1}\phi_1^{(m+1/2)}$

(c) $\phi_1^{(m+1)} \leftarrow \phi_1^{(m+1/2)} + \frac{1}{2}\epsilon D_\beta\left(\beta^{(m+1)}, \lambda^{(m+1)}, \delta^{(m+1)}\right)$

$\phi_2^{(m+1)} \leftarrow \phi_2^{(m+1/2)} + \frac{1}{2}\epsilon D_\lambda\left(\beta^{(m+1)}, \lambda^{(m+1)}, \delta^{(m+1)}\right)$

$\phi_3^{(m+1)} \leftarrow \phi_3^{(m+1/2)} + \frac{1}{2}\epsilon D_\delta\left(\beta^{(m+1)}, \lambda^{(m+1)}, \delta^{(m+1)}\right)$

Step 3: Accept or reject $(\beta^{(m+1)}, \lambda^{(m+1)}, \delta^{(m+1)}, \phi_1^{(m+1)}, \phi_2^{(m+1)}, \phi_3^{(m+1)})$ according to the Metropolis updates, set $m = m + 1$ and go to **Step 2**.

4.4 Applied study of a HMC methodology for a GARCH(1,1) model

In this section we apply the HMC methodology proposed in section 4.3. The HMC algorithms have been programmed in the `stan` (Stan Development Team (2014b)) software by means of `Rstan` package (Stan Development Team (2014a)). Codes are available in appendix A.7.

We first check the HMC methodology in order to estimate a GARCH model using simulated data. Then, we compare the obtained estimations with those obtained by using a MCMC and a data cloning approaches introduced in Section 3.2. Finally we apply the HMC methodology in a real data set and we compare results with those obtained with MCMC and data cloning approaches.

4.4.1 Simulation study of a HMC methodology with simulated GARCH data

In order to be able to compare the HMC results, with those that we obtained in Section 3.3, we use the same time series, which was simulated from a GARCH(1,1) model, with parameters $\beta = 3$, $\lambda = 0.7$ and $\delta = 0.1$, using `R` library `TSA` (Chan and Ripley (2012)). A plot of the simulated series may be observed in figure 3.1.

After simulating the series we apply the HMC method and we obtain the results included in Table 4.1. It is shown in columns, the real parameter values, the posterior means, the standard deviations and the HPD intervals for β, λ and δ. In order to make a comparison, we also include the estimations obtained with

Estimations using Hamiltonian Monte Carlo algorithm				
Parameter	Real Values	Estimator	S.D.	HPD 0.95
β	3	3.241	0.325	(2.715, 3.907)
λ	0.7	0.741	0.068	(0.632, 0.877)
δ	0.1	0.093	0.030	(0.047, 0.161)
Estimations using MCMC algorithm				
Parameter	Real Values	Estimator	S.D.	HPD 0.95
β	3	3.226	0.328	(2.612, 3.898)
λ	0.7	0.743	0.074	(0.606, 0.896)
δ	0.1	0.094	0.031	(0.042, 0.163)
Estimations using Data Cloning algorithm				
Parameter	Real Values	Estimator	S.D.	95% Confidence intervals
β	3	3.229	0.333	(2.589, 3.869)
λ	0.7	0.744	0.075	(0.597, 0.891)
δ	0.1	0.088	0.031	(0.027, 0.148)

Table 4.1: Estimates for GARCH(1,1) parameters using HMC, MCMC and data cloning

MCMC and data cloning approaches. It may be noticed that for all parameters the estimations are really close to the real values, and they are included in the HPD intervals. Furthermore, standard deviations of estimators are quite small in all cases, although they are a little bit smaller in the case of the HMC method.

The main improvement of the HMC methodology is the improvement of the computation times in contrast with the Gibbs sampler and Metropolis-Hasting algorithms, which sometimes take a long time zigging and zagging around the target distribution. On the contrary, the HMC methodology includes more efficient steps and it is significantly faster.

4.4.2 HMC methodology applied in real data modeled by a GARCH(1,1)

In this section we apply the HMC methodology on real data: weekly returns of the NASDAQ 100 stock index from January 2000 to November 2012. These are

the same data that we used to analyse the behaviour of the data cloning method in Section 3.2. In that section we also concluded that it could be modeled by a GARCH(1,1) model.

Estimations using Hamiltonian Monte Carlo algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	$8.19e$-05	$1.82e$-05	($5.41e$-05 ; $1.21e$-04)
λ	0.202	0.030	$(0.154; 0.267)$
δ	0.734	0.027	$(0.689; 0.788)$
Estimations using MCMC algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	$8.31e$-05	$1.94e$-05	($4.90e$-05 ; $1.25e$-04)
λ	0.2054	0.0324	$(0.147; 0.2748)$
δ	0.7324	0.0285	$(0.675; 0.7863)$
Estimations using Data Cloning algorithm			
Parameter	Estimated Value	S.D.	95% Confidence intervals
β	$7.82e$-05	$2.97e$-06	($4.13e$-05 ; $1.15e$-04)
λ	0.196	$5.01e$-03	$(0.134; 0.258)$
δ	0.741	$4.60e$-03	$(0.684; 0.798)$

Table 4.2: HMC, MCMC and Data Cloning estimations for GARCH(1,1) model for Nasdaq weekly returns data

In Table 4.2 it is shown in columns the posterior means of the parameters, the standard deviations and the HPD intervals for β, λ and δ. In order to make a comparison, we also include the estimations obtained with MCMC and data cloning approaches obtained in tables 3.9 and 3.8 respectively. It may be noticed that all values are similar among them and the series presents a persistent volatility, as we previously commented.

4.5 Hybrid Monte Carlo estimator for COGARCH model parameters

In this section we propose a HMC approach for a COGARCH(1,1) model assuming the approximation proposed by Maller et al. (2008) and described in section

2.2. As it was defined in (1.6), the parameters of interest of the model are β, η and φ which in the context of HMC models take the role of position variables. If we denote $\boldsymbol{\theta} = (\beta, \eta, \varphi)$, the potential energy is the derivative of the log-posterior density function

$$\frac{d \log P\left(\beta, \eta, \varphi | y\right)}{d\left(\beta, \eta, \varphi\right)} = \left(\frac{d \log P\left(\beta, \eta, \varphi | y\right)}{d\beta}, \frac{d \log P\left(\beta, \eta, \varphi | y\right)}{d\eta}, \frac{d \log P\left(\beta, \eta, \varphi | y\right)}{d\varphi} \right).$$

We set as prior distributions, the same vaguely-informative uniform distributions that we set in the cases of the MCMC and data cloning methodologies,

$$\beta \sim U(0, a)$$

$$\varphi \sim U(0, c)$$

$$\eta | \varphi \sim U(\varphi, b)$$

where a, c and b are positive real values and $b > c$.

Then, the posterior distribution is

$$\begin{aligned} & \pi(\beta, \varphi, \eta | y_1, \cdots, y_N) \propto L(\beta, \varphi, \eta) \pi(\beta) \pi(\varphi) \pi(\eta | \varphi) \\ \propto & \left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\rho_i^2}} \right) \exp\left\{ - \sum_{i=1}^{N} \frac{y_i^2}{2\rho_i^2} \right\} \cdot I_{(0,a)}(\beta) \cdot I_{(0,c)}(\varphi) \cdot I_{(\varphi,b)}(\eta | \varphi) \\ \propto & \left(\prod_{i=1}^{N} \frac{1}{\rho_i} \right) \exp\left\{ - \sum_{i=1}^{N} \frac{y_i^2}{2\rho_i^2} \right\} \cdot I_{(0,a)}(\beta) \cdot I_{(0,c)}(\varphi) \cdot I_{(\varphi,b)}(\eta | \varphi) \end{aligned}$$

where ρ_i is defined as

$$\rho_i^2 = \left(\sigma_{i-1}^2 - \frac{\beta}{\eta - \varphi} \right) \left(\frac{e^{(\eta - \varphi)\Delta t_i} - 1}{\eta - \varphi} \right) + \frac{\beta \Delta t_i}{\eta - \varphi}$$

and $\sigma_i^2 = \beta \Delta t_i + e^{-\eta \Delta t_i} \sigma_{i-1}^2 + \varphi e^{-\eta \Delta t_i} y_i^2$ and $\sigma_0^2 = \frac{\beta}{\eta - \varphi}$, with $\eta > \varphi$.

The momentum variables (ϕ_1, ϕ_2, ϕ_3) are auxiliary variables distributed as a multivariate normal distribution $\phi_j \sim N\left(0, I\right)$.

In order to calculate the corresponding potential energy, we define an intermediate function

$$g(\beta, \varphi, \eta) = -\sum_{i=1}^{N} \left(\log(\rho_i) + \frac{y_i^2}{2\rho_i^2} \right)$$

Let us denote the corresponding partial derivatives of $g(\beta, \varphi, \eta)$ with respect to (β, φ, η) as

$$\begin{aligned} D_\beta(\beta, \varphi, \eta) &= \frac{dg(\beta, \varphi, \eta)}{d\beta} \\ D_\varphi(\beta, \varphi, \eta) &= \frac{dg(\beta, \varphi, \eta)}{d\varphi} \\ D_\eta(\beta, \varphi, \eta) &= \frac{dg(\beta, \varphi, \eta)}{d\eta} \end{aligned}$$

For any t, the derivatives can be obtained by successive substitutions in $g(\beta, \varphi, \eta)$, although intractable and hugue polynomial expressions are obtained which should be better managed by numerical methods.

The HMC algorithm can be summarized as:

Step 1: Set initial values $(\beta^{(0)}, \eta^{(0)}, \varphi^{(0)})$ and $m = 0$.

Generate $(\phi_1^{(m)}, \phi_2^{(m)}, \phi_3^{(m)})$ from $N(0, I)$.

Step 2: Generate proposed values for $(\theta^{(m+1)}, \phi^{(m+1)})$ acording to leapfrog method, as follows

(a) $\phi_1^{(m+1/2)} \leftarrow \phi_1^{(m)} + \frac{1}{2}\epsilon D_\beta(\beta^{(m)}, \eta^{(m)}, \varphi^{(m)})$

$\phi_2^{(m+1/2)} \leftarrow \phi_2^{(m)} + \frac{1}{2}\epsilon D_\eta(\beta^{(m)}, \eta^{(m)}, \varphi^{(m)})$

$\phi_3^{(m+1/2)} \leftarrow \phi_3^{(m)} + \frac{1}{2}\epsilon D_\varphi(\beta^{(m)}, \eta^{(m)}, \varphi^{(m)})$

(b) $\beta^{(m+1)} \leftarrow \beta^{(m)} + \epsilon M^{-1}\phi_1^{(m+1/2)}$

$\eta^{(m+1)} \leftarrow \eta^{(m)} + \epsilon M^{-1}\phi_2^{(m+1/2)}$

$\varphi^{(m+1)} \leftarrow \varphi^{(m)} + \epsilon M^{-1}\phi_1^{(m+1/2)}$

(c) $\phi_1^{(m+1)} \leftarrow \phi_1^{(m+1/2)} + \frac{1}{2}\epsilon D_\beta(\beta^{(m+1)}, \eta^{(m+1)}, \varphi^{(m+1)})$

$\phi_2^{(m+1)} \leftarrow \phi_2^{(m+1/2)} + \frac{1}{2}\epsilon D_\eta(\beta^{(m+1)}, \eta^{(m+1)}, \varphi^{(m+1)})$

$\phi_3^{(m+1)} \leftarrow \phi_3^{(m+1/2)} + \frac{1}{2}\epsilon D_\varphi(\beta^{(m+1)}, \eta^{(m+1)}, \varphi^{(m+1)})$

Step 3: Accept or reject $(\theta^{(m+1)}, \eta^{(m+1)}, \phi^{(m+1)})$ according to the Metropolis updates, set $m = m + 1$ and go to **Step 2**.

4.6 Applied study of a HMC methodology for a COGARCH(1,1) model

In this section we apply the HMC methodology proposed in section 4.5. The HMC algorithms have been programmed in the `stan` (Stan Development Team (2014b)) software by means of `Rstan` package (Stan Development Team (2014a)). Codes are available in appendix A.8.

We first check the HMC methodology in order to estimate a COGARCH model using simulated data. Then, we compare the obtained estimations with those obtained by using a MCMC and a data cloning approaches introduced in Sections 2.3 and 3.4. Finally we apply the HMC methodology in a real data set and we compare results with those obtained with MCMC and data cloning approaches.

4.6.1 Simulation study of a HMC methodology with simulated COGARCH data

In order to be able to compare the HMC results, with those that we obtained in Section 3.5.1, we use the same time series, which was simulated from a COGARCH(1,1) model, with parameters $\beta = 0.25$, $\eta = 0.35$ and $\varphi = 0.02$,

using the code from Appendix A.1. A plot of the simulated series may be observed in figure 2.1. By the other hand, codes about estimations of the HMC method can be found in Appendix A.8.

After simulating the series we apply the HMC method and we obtain the results included in Table 4.3. It is shown in columns, the real parameter values, the posterior means, the standard deviations and the HPD intervals for β, η and φ. In order to make a comparison, we also include the estimations obtained with MCMC and data cloning approaches in tables 2.1 and 3.11. It may be noticed that for all parameters the estimations are close to the real values (remarkably η and φ), and all real values are included in the HPD intervals. Furthermore, standard deviations of estimators are small in all cases.

Estimations using Hamiltonian Monte Carlo algorithm				
Parameter	Real Value	Estimator	S.D.	HPD 0.95
β	0.25	0.542	0.202	(0.162, 0.948)
η	0.35	0.323	0.144	(0.088, 0.367)
φ	0.02	0.027	0.014	(0.002, 0.048)
Estimations using MCMC algorithm				
β	0.25	0.427	0.158	(0.138 ; 0.747)
η	0.35	0.301	0.132	(0.092 ; 0.600)
φ	0.02	0.029	0.014	(0.002 ; 0.049)
Estimations using Data Cloning algorithm				
β	0.25	0.26795	0.21116	(0, 0.682)
η	0.35	0.18782	0.14597	(0, 0.474)
φ	0.02	0.04481	0.02122	(0.003, 0.086)

Table 4.3: Estimation for COGARCH(1,1) parameters using HMC

In general, in COGARCH(1,1) models, because the wild behaviour of its trajectories, the computation time of MCMC methods is quite large. The advantage of the HMC methodology based on calculating efficient steps allows to improve quite enough the computation of the posterior distributions of the parameters.

4.6.2 HMC methodology applied in real data modeled by a COGARCH(1,1)

In this section we apply the HMC methodology on real data: daily returns of the Nasdaq index from January 2008 to December 2012. These are the same data that we used to analyse the behaviour of the data cloning method of Section 3.5.2.

Table 4.4 shows the estimations for parameters of COGARCH(1,1) model for the series of daily Nasdaq returns. It also include the obtained results using MCMC and Data Cloning algorithms from Tables 3.15 and 3.14 respectively. It could be observed that the estimations are quite similar to the ones obtained using data cloning or MCMC as expected. The reduction in the computation times is again the main advantage of this algorithm, which, as we mention is specially important when estimating COGARCH parameters, because MCMC takes a specially large amount of time in arising convergence.

In Table 4.4 it is shown in columns the posterior means of the parameters, the standard deviations and the HPD intervals for β, η and φ. In order to make a comparison, we also include the estimations obtained with MCMC and data cloning approaches obtained in tables 3.15 and 3.14 respectively. It may be noticed that all values are similar among them. Nevertheless, the HMC algorithm is faster than the other ones.

Estimations using a Hamiltonian Monte Carlo algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	9.637	2.064	$(6.155, 14.289)$
η	0.053	0.004	$(0.045, 0.062)$
φ	0.046	0.002	$(0.038, 0.049)$
Estimations using MCMC algorithm			
β	9.661	2.113	$(6.140; 14.414)$
η	0.054	0.0043	$(0.045; 0.063)$
φ	0.047	0.0028	$(0.039; 0.050)$
Estimations using Data Cloning algorithm			
β	9.041	0.289	$(5.457; 12.626)$
η	0.055	0.00042	$(0.050; 0.061)$
φ	0.05	0.00015	$(0.048; 0.052)$

Table 4.4: HMC for COGARCH(1,1) model for Nasdaq daily returns data

Chapter 5

Approximate Bayesian Computation estimation for GARCH and COGARCH model parameters

5.1 Introduction to Approximate Bayesian Computation Methods

The Approximate Bayesian Computation (ABC) methodology is a relative new technique, which has arisen to cope with the estimation of parameters in models where the likelihood function is intractable.

When the likelihood function is not available for mathematical or computational reasons, frequentist and also Bayesian methodologies may have problems to estimate the model parameters. Hierarchical models may be examples of these situations, because they present sometimes partly conjugate prior distributions and it is not always possible to implement Metropolis-Hasting algorithms. Moreover, when increasing parameter dimensions, the convergence of the algorithms becomes more difficult and they arise poorer estimations.

In other cases the likelihood function depends on an unknown normalizing constant. This is a common situation in spatially correlated data and a possible solution in these situations is to use pseudo-samples as Møller et al. (2006) proposed. But there is dependency on a pseudo-target distribution which makes the algorithm difficult to calibrate.

ABC methods address a large amount of situations in which the likelihood function is not completely known and the exact simulation from the posterior distributions is not possible. In some of these cases they can be approximated by using Laplace approximations (Tierney and Kadane (1986)), but they require to know, at least analytically, the posterior distribution. Variational Bayes solutions (Jaakkola and Jordan (2000)) can be also applied in some situations, but they consist on replacing the true model by a pseudo-model, usually simpler, hence it misses information of the original model features.

In this chapter we introduce estimators for GARCH an COGARCH model parameters using ABC methods. The proposed algorithms provide good enough estimations for GARCH model parameters but especially for COGARCH model parameters, because this model produce sample paths with a specially wild behaviour. Furthermore, the existing estimators for parameters are based on a pseudo-likelihood function. In GARCH models this pseudo-likelihood function arise good enough estimations, but in COGARCH models pseudo maximum likelihood estimators do not work quite accurately until now. In the next section we describe the Approximate Bayesian Computation framework and then we introduce the ABC estimators and analyse their quality.

5.2 Description of Approximate Bayesian Computation Algorithms

ABC methods can be seen as a solution to find estimates in models where the likelihood function is not completely known or it is intractable, although it is possible to simulate from it.

The first ABC Method was pointed out by Rubin (1984) who wrote a seminal paper which included the first description of an ABC algorithm. After that Tavare et al. (1997) described the proper ABC algorithm which was an acceptance or rejection method in which parameters $\boldsymbol{\theta}$ were generated from a prior distribution $\pi(\boldsymbol{\theta})$. Then it is included an acceptance-rejection condition for the simulated values to be compared with the original data.

If we denote as y the original sample which takes values in a finite set D, the original ABC algorithm from Tavare et al. (1997) can be summarised as:

Original ABC Algorithm

for $t = 1$ to N **do**

repeat

Generate $\boldsymbol{\theta}'$ from the prior distribution $\pi(\boldsymbol{\theta})$

Generate z from the likelihood function $f(\cdot|\boldsymbol{\theta}')$

until $z = y$

set $\boldsymbol{\theta}_t$=$\boldsymbol{\theta}'$

end for

The outcome of this algorithm is an iid sample from the posterior distribution of $\boldsymbol{\theta}$, and

$$f(\boldsymbol{\theta}_t) \propto \sum_{z \in D} \pi(\boldsymbol{\theta}_t) f(z|\boldsymbol{\theta}_t) I_y(z) = \pi(\boldsymbol{\theta}_t) f(z|\boldsymbol{\theta}_t) \propto \pi(\boldsymbol{\theta}_t|y).$$

Although Rubin (1984) does not recommend to use this algorithm when the likelihood function is unknown, he agrees that it allows to understand the posterior distribution in an intuitive way, based on a frequentist framework. We should remark that Rubin (1984) does not specify the concept of proximity between the original data and the simulated sample, and he only points out that they have to match.

Subsequently Pritchard et al. (1999) extended the previous algorithm to be used in continuous sample spaces. In this way, he proposed the first practical ABC algorithm which included some instrumental statistics. He defined a summary statistic η (a function on D but usually not a sufficient statistic); also ρ, a distance between η applied on the estimated and real data, and a tolerance level ϵ.

Then, the algorithm can be described as:

ABC algorithm

for $i = 1$ to N **do**

repeat

Generate $\boldsymbol{\theta}'$ from the prior distribution $\pi(\cdot)$

Generate z from the likelihood $f(\cdot|\boldsymbol{\theta}')$

until $\rho\{\eta(z), \eta(y)\} \leq \epsilon$

set $\boldsymbol{\theta}_i$=$\boldsymbol{\theta}'$

end for

The algorithm of Pritchard et al. (1999) obtains samples of the joint distribution of $\boldsymbol{\theta}$ and z, by means of the marginal distribution of z

$$\pi_{\varepsilon}(\boldsymbol{\theta}, z|y) = \frac{\pi(\boldsymbol{\theta}) f(z|\boldsymbol{\theta}) I_{A_{\epsilon,y}}(z)}{\int_{A_{\epsilon,y}\times\theta} \pi(\boldsymbol{\theta}) f(z|\boldsymbol{\theta}) dz d\boldsymbol{\theta}}, \tag{5.1}$$

where $I(\cdot)$ is an indicator function and

$$A_{\epsilon,y} = \{z \in D | \rho\{\eta(z), \eta(y)\} \leq \epsilon\}.$$

The basic idea is, instead of a direct comparison between the real data and the simulated samples, to use a statistic η and fix a small enough distance between them, given by ϵ. Then, the algorithm should render good enough approximations to the posterior distribution

$$\pi_{\epsilon}(\boldsymbol{\theta}|y) = \int \pi_{\epsilon}(\boldsymbol{\theta}, z|y) dz \approx \pi(\boldsymbol{\theta}|y).$$

Obviously, when the tolerance level becomes smaller, better approximations to the posterior distribution are found, but with a higher computation cost.

Nevertheless, simulating directly from the prior distribution may be inefficient in practice, since there is not a stage which leads to proposed values, so it makes difficult to obtain values which are located in regions with a low posterior probability. In order to solve this problem, Marjoram et al. (2003) proposed the MCMC-ABC algorithm which targets the posterior distribution defined in 5.1.

The algorithm can be described as

MCMC-ABC algorithm

Obtain initial values for $(\boldsymbol{\theta}^{(0)}, z^{(0)})$ from the ABC target distribution $\pi_\epsilon(\boldsymbol{\theta}, z|y)$ by using the ABC algorithm.

for $t = 1$ to N **do**

Generate $\boldsymbol{\theta}'$ from a Markov kernel $q\left(\cdot|\boldsymbol{\theta}^{(t-1)}\right)$

Generate $z\prime$ from the likelihood $f(\cdot|\boldsymbol{\theta}')$

Generate u from $U[0,1]$

if $u \leq \frac{\pi(\theta')q\left(\theta^{(t-1)}|\theta'\right)}{\pi(\theta^{(t-1)})q\left(\theta'|\theta^{(t-1)}\right)}$ and $\rho\left\{\eta(z'), \eta(y)\right\} \leq \epsilon$ **then**

set $(\boldsymbol{\theta}^{(t)}, z^{(t)})$=$(\boldsymbol{\theta}', z')$

else

set $(\boldsymbol{\theta}^{(t)}, z^{(t)})$=$(\boldsymbol{\theta}^{(t-1)}, z^{(t-1)})$

end if

end for

The ABC requirement about not using the likelihood function is satisfied by

the MCMC-ABC algorithm, because it does not require the likelihood function to obtain the acceptance probability. This algorithm is also a MCMC method which targets the exact stationary distribution $\pi_\epsilon(\boldsymbol{\theta}, z|y)$.

The acceptance probability for this algorithm can be obtained as $q\left(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(t-1)}\right)$

$$\frac{\pi_\epsilon(\boldsymbol{\theta}', z'|y)}{\pi_\epsilon(\boldsymbol{\theta}^{(t-1)}, z^{(t-1)}|y)} \times \frac{q\left(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\theta}'\right) f\left(z^{(t-1)}|\boldsymbol{\theta}^{(t-1)}\right)}{q\left(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(t-1)}\right) f(z'|\boldsymbol{\theta}')} =$$

$$\frac{\pi(\boldsymbol{\theta}')f(z'|\boldsymbol{\theta}')I_{A_{\epsilon,y}(z')}}{\pi(\boldsymbol{\theta}^{(t-1)})f(z^{(t-1)}|\boldsymbol{\theta}^{(t-1)})I_{A_{\epsilon,y}(z^{(t-1)})}} \times \frac{q\left(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\theta}'\right) f\left(z^{(t-1)}|\boldsymbol{\theta}^{(t-1)}\right)}{q\left(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(t-1)}\right) f(z'|\boldsymbol{\theta}')} =$$

$$\frac{\pi(\boldsymbol{\theta}')q\left(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\theta}'\right)}{\pi(\boldsymbol{\theta}^{(t-1)})q\left(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(t-1)}\right)} I_{A_{\epsilon,y}(z')}.$$

As an ergodic Markov chain forgets its initial state, the first step which involves the use of an ABC algorithm can be omitted in order to reduce the computational cost. Nevertheless it is better to run the MCMC-ABC algorithm so many iterations as to achieve convergence, so the improvement in the computational cost will be not very significant in practice.

The MCMC-ABC algorithm also depends on the tuning parameters, as the summary statistics η, the distance ρ and the tolerance ϵ which have to be set by the researcher before running the algorithm, and the handy selection of them improves results in this methodology. The calibration and setting of the tuning parameters is an open issue.

McKinley et al. (2009) carried out an empirical analysis to test different strategies to select the tolerance level ϵ, the distance ρ and the summary statistics η. They remarked the following conclusions:

(i) Simulation of several data sets from the same proposal parameters does not

improve the approximation of the posterior distributing when using ABC algorithms.

(*ii*) The tolerance level ϵ does not affect significantly to the results.

(*iii*) The selection of the summary statistics and their distance is essential for obtaining accurate approximations to the posterior distribution.

Therefore, the election of η and ρ is the most important step. Joyce and Marjoram (2008) consider the usual summary statistics available and suggest a sequential inclusion of these in the ABC algorithm. They propose to include a new statistic related to η after assessing it, by using a likelihood ratio test, but they do not take into account the sequential nature of the test.

In order to implement an ABC algorithm to estimate GARCH and COGARCH model parameters, we have studied the effect of changing the parameters in the sample paths, and the most appropriated set of summary statistics. We have tested different summary statistics, and we have concluded that in order to find good enough results in a reasonable computation time, it is better to limit prior distributions around the maximum likelihood or the moments estimates of parameters. We have also considered different distances ρ and we have concluded that the euclidean distance is the most appropriate one. In next sections we describe the proposed algorithm to estimate GARCH and COGARCH model parameters.

5.3 Approximate Bayesian Computation estimation for GARCH model parameters

In this section we deal with estimates of the GARCH model parameters based on an ABC methodology.

We refer to section 1.3 for the definition of a GARCH(1,1) model,

$$\begin{aligned} Y_n &= \sigma_n \epsilon_n, \\ \sigma_n^2 &= \beta + \lambda Y_{n-1}^2 + \delta \sigma_{n-1}^2, \end{aligned}$$

for $n \geq 0$ where $\beta > 0$, $\lambda \geq 0$ and $\delta \geq 0$, and ϵ_n are a sequence of random variables, independently and identically normally distributed.

As usual, the model parameters to estimate are β, λ and δ and it is necessary assign prior distributions for them. The parameters space is given by

$$\Phi \subset \mathbb{R}^3 \times (0, \infty) \times (0, 2e^{\gamma}) \times (0, 1)$$

Assuming the restrictions addressed in proposition 9, we use similar weak informative uniform distributions as in the case of the Data Cloning and the Hamiltonian Monte Carlo techniques (see sections 3.2 and 4.3)

$$\begin{aligned} \beta &\sim U(0, d) \\ \lambda &\sim U(0, 2e^{\gamma}) \\ \delta &\sim U(0, 1) \end{aligned}$$

where d is any positive value, and γ is the Euler-Mascheroni constant.

Then, the joint posterior distribution is

$$\pi(\beta, \lambda, \delta | Y) \propto \prod_{n=1}^{N} \left(\sqrt{2\pi\sigma_n^2} \right)^{-1} \exp\left(-\frac{y_n^2}{2\sigma_n^2} \right) \cdot I_{(0,d)}(\beta) \cdot I_{(0,2e^{\gamma})}(\lambda) \cdot I_{(0,1)}(\delta)$$

where $\sigma_n^2 = \beta + \lambda Y_{n-1}^2 + \delta \sigma_{n-1}^2$.

Before describing the algorithm, it is necessary to set the summary statistics $\eta(Y)$ that will be used when running the ABC algorithm. In these sense, we undertook empirical studies to conclude that it is not necessary to use cumbersome statistics if the prior distributions of parameters are restricted in some

way. Accordingly, we use as statistics the sample mean, the sample variance and the median absolute deviation. And in order to fix the prior distributions of the parameters, we use uniform distributions whose supports are around the quasi-maximum likelihood estimates ($QMLE$) of the parameters.

The quasi-likelihood function is

$$L_N = L_N(\beta, \lambda, \delta, Y) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{Y_n^2}{2\sigma_n^2}\right),$$

and the QMLE of (β, λ, δ) is obtained as

$$\hat{\theta}_N = \arg\max_{\theta \in \Phi} L_N(\beta, \lambda, \delta, Y).$$

According to Francq et al. (2004), the maximum of $L_N(\beta, \lambda, \delta, Y)$ is equivalent to the minimum of

$$I_N(\beta, \lambda, \delta, Y) = N^{-1} \sum_{n=1}^{N} \frac{Y_n^2}{\sigma_n^2} + \log \sigma_n^2.$$

With respect to an ABC algorithm, the mean, the variance and the median absolute deviations will be used as summary statistics, defining the function $\eta(Y)$ as

$$\eta(Y) = (\bar{Y}, Var[Y], mad[Y]).$$

being median absolute deviations described as

$$mad = median_n\left(|Y_n - median_n(Y_n)|\right)$$

Then, the algorithm can be described as follows:

for $m = 1$ **to** N

Step 1: Generate $(\beta^*, \lambda^*, \delta^*)$ from the prior distributions.

Step 2: Generate Y^* from the quasi-likelihood function, L_N

Step 3: **If** $\sqrt{\left(\bar{Y}-\bar{Y^*}\right)^2+\left(Var\left[Y\right]-Var\left[Y^*\right]\right)^2+\left(mad\left[Y\right]-mad\left[Y^*\right]\right)^2}<\epsilon$

then

set $(\beta^{(m)},\lambda^{(m)},\delta^{(m)},Y^{(m)})$=$(\beta_0^*,\lambda_1^*,\delta^*,Y^*)$ and $m=m+1$

else

go to Step 1

end if

end for

Alternatively, other solution may be using a MCMC-ABC algorithm, which may be found more appropriated considering that the MCMC methodology works properly in the GARCH models framework.

The MCMC-ABC algorithm can be described as follows:

Step 1: Set initial values $(\beta^{(0)},\lambda^{(0)},\delta^{(0)})$.

for $t=1$ **to** N

Step 2: Generate $(\beta^*,\lambda^*,\delta^*)$ from a Markov kernel $q(\beta,\lambda,\delta|\beta^{(m-1)},\lambda^{(m-1)},\delta^{(m-1)})$.

Step 3: $Y_n^*\leftarrow\sigma_n\epsilon_n$.

being $\sigma_n^2=\beta^*+\lambda^*Y_{n-1}^2+\delta^*\sigma_{n-1}^2$.

Step 4: Generate u from $U[0,1]$.

Step 5: **If** $u\leq\frac{\pi(\beta^*,\lambda^*,\delta^*|Y)q(\beta^{(m-1)},\lambda^{(m-1)},\delta^{(m-1)}|\beta^*,\lambda^*,\delta^*)}{\pi(\beta^{(m-1)},\lambda^{(m-1)},\delta^{(m-1)}|Y))q(\beta^*,\lambda^*,\delta^*|\beta^{(m-1)},\lambda^{(m-1)},\delta^{(m-1)})}$.

and $\rho\left\{\eta(Y^*),\eta(Y)\right\}\leq\epsilon$ **then**

set $(\beta^{(m)},\lambda^{(m)},\delta^{(m)},Y^{(m)})$=$(\beta^*,\lambda^*,\delta^*,Y^*)$

else

set $(\beta^{(m)}, \lambda^{(m)}, \delta^{(m)}, Y^{(m)})$=$(\beta^{(m-1)}, \lambda^{(m-1)}, \delta^{(m-1)}, Y^{(m-1)})$

end if

end for

Both algorithms have been programmed in `R` (R Core Team (2012)) using the package `fGarch` (Wuertz et al. (2013)) to obtain the QMLE estimates, and `EasyABC` (Jabot et al. (2014)) package to apply the ABC algorithm. Codes are available in appendix A.9 and A.10.

5.4 Application of the ABC methodology for GARCH models

In this section, we undertake first a simulation study to check the estimators obtained by an ABC methodology in a GARCH(1,1) model. Then, the ABC method is applied over Nasdaq index returns (from January 2008 to December 2012) and we compare the obtained results with those of the former algorithms proposed in this thesis.

5.4.1 Simulation study of ABC with simulated GARCH data

As in previous sections, we use the simulation of the 1000 observations from a GARCH(1,1) model with parameters $\beta = 3$, $\lambda = 0.7$ and $\delta = 0.1$, by means of the `R` library `TSA` (Chan and Ripley (2012)). Simulated data are plotted in 3.1.

We use uniform prior distributions for the parameters whose supports are around the quasi-maximum likelihood estimates ($QMLE$) as discussed in section 5.3.

We take 500000 simulations in the ABC algorithm in order to compare the target statistics (mean, variance and median absolute distances) with the values of the original series. Using a tolerance level of 0.005, 2507 values for parameters have been accepted in order to determine the posterior distributions for all parameters.

Estimations using ABC algorithm				
Parameter	Real Values	Estimations	S.D.	HPD 0.95
β	3	2.894	0.115	(2.707, 3.084)
λ	0.7	0.686	0.115	(0.497, 0.877)
δ	0.1	0.011	0.067	(0.005, 0.222)
Estimations using Hamiltonian Monte Carlo algorithm				
Parameter	Real Values	Estimations	S.D.	HPD 0.95
β	3	3.241	0.325	(2.715, 3.907)
λ	0.7	0.741	0.068	(0.632, 0.877)
δ	0.1	0.093	0.030	(0.047, 0.161)
Estimations using Data Cloning algorithm				
Parameter	Real Values	Estimations	S.D.	95% Confidence intervals
β	3	3.229	0.333	(2.589, 3.869)
λ	0.7	0.744	0.075	(0.597, 0.891)
δ	0.1	0.088	0.031	(0.027, 0.148)
Estimations using MCMC algorithm				
Parameter	Real Values	Estimations	S.D.	HPD 0.95
β	3	3.226	0.328	(2.612, 3.898)
λ	0.7	0.743	0.074	(0.606, 0.896)
δ	0.1	0.094	0.031	(0.042, 0.163)

Table 5.1: Estimation for GARCH(1,1) parameters using ABC, HMC, MCMC and Data Cloning

Table 5.1 shows the estimations obtained by using an ABC algorithm over the simulated GARCH sample path. For comparison tasks, it also contains results obtained with Hamiltonian Monte Carlo, Data cloning and MCMC algorithms. For all of them the table shows by columns, the real values used to simulate the sample path, the estimates, the obtained standard deviations and HPD intervals (confidence intervals in data cloning framework).

In case of ABC, it can be noted that the obtained estimates are quite close to the real values of parameters, like by using the other Bayesian approaches. And all HPD interval include the real parameter values, so we can conclude that ABC approach renders accurate results.

Similar results may have been obtained without including restrictions in the prior distributions, but in order to obtain 2507 accepted values with a small tolerance level, it would necessary using much more than 500000 simulations, increasing significantly the computation time.

By the other hand, if we apply an ABC-MCMC algorithm over the same series, similar results are obtained as it is shown in table 5.2.

The algorithm can be tried with a larger number of simulations, but it is known that a larger number of iterations it does not improve significantly the results and constitute an added cost of highly increasing computing times.

Estimations using ABC-MCMC algorithm				
Parameter	Real Values	Estimator	S.D.	HPD 0.95
β	3	2.886	0.108	(2.711, 3.078)
λ	0.7	0.745	0.033	(0.663, 0.784)
δ	0.1	0.088	0.029	(0.048, 0.127)

Table 5.2: Estimation for GARCH(1,1) parameters using ABC-MCMC

Table 5.2 shows the real value of the parameters, the obtained estimates by using the ABC-MCMC approach, their standard errors and the HPD intervals for β, λ and δ. Comparing results with those shown in table 5.1, it can be observed that the ABC-MCMC method provides estimates with a significantly smaller standard deviations than in the case of MCMC, data cloning or HMC algorithms, for all parameters except for β. And these standard deviation can be reduced by increasing the number of simulations. Even the standard deviations

for all parameters are smaller than in the case of ABC estimations. Consequently, HPD intervals are smaller for β and λ and similar for δ. This points out that the ABC-MCMC estimator is more efficient than the rest of Bayesian estimators. According to this could be affirm that ABC-MCMC algorithm provides the more efficient estimators.

5.4.2 ABC estimations for real data modeled by GARCH

In this section we apply the ABC and ABC-MCMC methodologies on real data: the series of Nasdaq 100 weekly return from January 2000 to November 2012. As we analysed in section 3.3.2, this series includes GARCH effects and the best model to apply, according to the Akaike and Schwarz criteria is a GARCH(1,1).

Estimations using ABC algorithm			
Parameter	Estimations	S.D.	HPD 0.95
β	0.100	0.057	(0.004, 0.195)
λ	0.152	0.088	(0.007, 0.297)
δ	0.839	0.0839	(0.687, 0.992)
Estimations using Hamiltonian Monte Carlo algorithm			
Parameter	Estimations	S.D.	HPD 0.95
β	$8.19e$-05	$1.82e$-05	($5.41e$-05 ; $1.21e$-04)
λ	0.202	0.030	$(0.154; 0.267)$
δ	0.734	0.027	$(0.689; 0.788)$
Estimations using Data Cloning algorithm			
Parameter	Estimations	S.D.	95% confidence intervals
β	$7.82e$-05	$2.97e$-06	($4.13e$-05 ; $1.15e$-04)
λ	0.196	$5.01e$-03	$(0.134; 0.258)$
δ	0.741	$4.60e$-03	$(0.684; 0.798)$
Estimations using MCMC algorithm			
Parameter	Estimations	S.D.	HPD 0.95
β	$8.31e$-05	$1.94e$-05	($4.90e$-05 ; $1.25e$-04)
λ	0.2054	0.0324	$(0.147; 0.2748)$
δ	0.7324	0.0285	$(0.675; 0.7863)$

Table 5.3: ABC, HMC, MCMC and Data Cloning estimations for GARCH(1,1) model for Nasdaq weekly returns data

Table 5.3 shows the estimated values, standard deviations and HPD intervals of the model parameters, by using ABC, HMC, Data Cloning and MCMC methods.

We consider for the ABC approach 500000 simulations, such that 2494 of them are accepted, with a tolerance level of 0.005. It can be observed that estimates are similar to the obtained with the rest of Bayesian methodologies.

Estimations using ABC-MCMC algorithm			
Parameter	Estimator	S.D.	HPD 0.95
β	0.038	0.025	(0.002, 0.094)
λ	0.032	0.021	(0.002, 0.084)
δ	0.759	0.059	(0.683, 0.899)

Table 5.4: ABC-MCMC estimations for GARCH(1,1) model for Nasdaq weekly returns data

The results of the ABC-MCMC methodology are shown in table 5.4, which includes the estimates, the standard deviations and the HPD intervals. We consider 400000 simulations and then, similar estimates as in case of the other Bayesian techniques are found as comparing with table 5.3.

Anyway, as the frequentist approach and the standard Bayesian techniques obtain accurate results, the introduction of the ABC approach may be taken into consideration as a former toy example in order to introduce the methodology in more complex models as COGARCH ones.

5.5 Approximate Bayesian Computation for COGARCH model parameters

In this section we propose an ABC approach for a COGARCH(1,1) model assuming the approximation proposed by Maller et al. (2008) and described in Section

2.2.

Let us recall the definition of a COGARCH (1,1) process from (1.6), in which we defined G_t to be a COGARCH(1,1) process when

$$dG_t = \sigma_t dL_t$$

where $t > 0$, $G_0 = 0$, β, η and φ are the model parameters with $\beta > 0$, $\eta > 0$ and $\varphi \geq 0$ and the variance process is defined as

$$d\sigma^2_{t+} = \beta dt - \eta\sigma^2_t dt + \varphi\sigma^2_t d[L, L]_t$$

where $[L, L]_t$ is the quadratic variation of the Lèvy process.

The same prior distributions than in previous chapters are going to be used,

$$\beta \sim U(0, a)$$

$$\varphi \sim U(0, c)$$

$$\eta|\varphi \sim U(\varphi, b)$$

where a, c and b are positive real values and $b > c$.

Before describing the algorithm, it is necessary to set the summary statistics $\eta(Y)$ that will be used when running the ABC algorithm.

In section 1.4.3, when we analysed the behaviour of the COGARCH(1,1) process in relation with the parameters change, we concluded that the minimum value of volatility has an important influence on the parameters, especially on η and β. In this way there is a strong relationship between these parameters and the volatility lower bound, therefore we introduce this statistic in the statistics function $\eta(Y)$.

It is also known that parameter φ is the responsible of the process jumps, so

we include the information of their magnitude in terms of the difference between the maximum and the minimum observed volatility.

Another relevant statistics is the covariance between the squared of the series values in two time points (which we will consider as consecutive). This covariance is used when defining the moment estimator (see 2.4) and it constitutes one of the stylized facts analysed by Mandelbrot (1963) to define the characteristic of financial series, namely the "Squared price returns autocorrelated". It is known that financial series present no autocorrelated returns, but there is a high correlation between the squared returns. Consequently, we include in $\eta(Y)$ the correlation coefficient of the squared process values. Finally, we include in the set of statistics $\eta(Y)$ the sample variance and median absolute deviations of the sample path.

In the same way as in the GARCH model case, in order to reduce computation times we restrict the prior distributions of parameters in some way. Accordingly, we use as a simple approach uniform distributions whose supports are around the the moment estimator defined by Haug et al. (2005).

The ABC algorithm to estimate the COGARCH model parameters (η, β, φ) can be described as follows:

$m = 1$

while $m \leq N$

Step 1: Generate $(\eta^*, \beta^*, \varphi^*)$ from the prior distributions.

Step 2: Generate G^* from the quasi-likelihood function, L_n

Step 3: **If** $\sqrt{(}\,(MinVol(G) - MinVol(G^*))^2 + (Range(Vol(G)) - Range(Vol(G^*)))^2$

$+ \left(\rho\left(G_t^2, G_{t-1}^2\right) - \rho\left((G^*)_t^2, (G^*)_{t-1}^2\right)\right)^2 +$

$+ (Var[G] - Var[G^*])^2 + (mad[G] - mad[G^*])^2) < \epsilon$ **then**

set $(\eta^{(m)}, \beta^{(m)}, \varphi^{(m)}, G^{(m)})$=$(\eta^*, \beta^*, \varphi^*, G^*)$ and $m = m + 1$.

else

go to Step 1

end if

end while

The algorithm has been programmed in `R` (R Core Team (2012)) using the package `COGARCH` (Bibbona et al. (2014)) in order to calculate the moment estimators to restrict the prior distributions, and the package `EasyABC` (Jabot et al. (2014)) to apply the ABC algorithm. Codes are available in appendix A.11.

5.6 Simulation study of ABC estimator for COGARCH(1,1) model parameters

In this section, we undertake first a simulation study to check the estimators obtained by an ABC methodology in a COGARCH(1,1) model. Then, the ABC method described in section 5.5 is applied over Nasdaq index returns (from January 2008 to December 2012) and we compare the obtained results with those of the former algorithms proposed in this thesis in sections 2.3, 3.4 and 4.5.

5.6.1 Simulation study of ABC with simulated COGARCH(1,1) data

In this simulation study, we have simulated series of 100 data points with parameters $\beta = 0.25$, $\eta = 0.35$ and $\varphi = 0.02$ (see the `R` codes in appendix A.1). Figure

2.1 shows the obtained sample path used to analyse and discuss the behaviour of the other Bayesian approaches taken in this thesis.

Estimations using ABC algorithm				
Parameter	Real Value	Estimator	S.D.	HPD 0.95
β	0.25	0.336	0.121	(0.114, 0.502)
η	0.35	0.305	0.146	(0.096, 0.582)
φ	0.02	0.025	0.014	(0.002, 0.047)
Estimations using Hamiltonian Monte Carlo algorithm				
Parameter	Real Value	Estimator	S.D.	HPD 0.95
β	0.25	0.542	0.202	(0.162, 0.948)
η	0.35	0.323	0.144	(0.088, 0.367)
φ	0.02	0.027	0.014	(0.002, 0.048)
Estimations using Data Cloning algorithm				
Parameter	Real Value	Estimator	S.D.	95% Confidence intervals
β	0.25	0.26795	0.21116	(0, 0.682)
η	0.35	0.18782	0.14597	(0, 0.474)
φ	0.02	0.04481	0.02122	(0.003, 0.086)
Estimations using MCMC algorithm				
Parameter	Real Value	Estimator	S.D.	HPD 0.95
β	0.25	0.427	0.158	(0.138 ; 0.747)
η	0.35	0.301	0.132	(0.092 ; 0.600)
φ	0.02	0.029	0.014	(0.002 ; 0.049)

Table 5.5: Estimation for COGARCH(1,1) parameters using ABC algorithm

The results obtained by applying ABC algorithm over the simulated data set are shown in table 5.5. It shows by columns the real values of parameters, the obtained estimates, the standard deviation and the HPD intervals. We have used 500000 simulations with a tolerance level of 0.01, in such a way that 5000 sets of parameters have been accepted as an obtained sample from the posterior distribution. We have undertaken parallel computing in codes, by means of Analytics and Weston (2014) with a significant diminishing computing times in a multicore machine.

In the case of the ABC approach, it can be noted that the estimations for η, β

and φ are quite close to real values and all HPD intervals include them. Standard deviations for parameter β is smaller than those obtained with the rest of Bayesian approaches proposed in this thesis, whose results are also included in this table. Consequently the HPD interval is also smaller. Although for parameter η the ABC approaches obtains the larger standard deviation in relation with the rest of methods.

The advantage of the ABC approach is that it is a simulation-based method, where it is not necessary to deal with indirect quasi-maximum likelihood techniques or approximate moment methods. Although the computational burden may be huge if the support of the prior distributions is not restricted somehow.

5.6.2 ABC estimations for real data using a COGARCH(1,1)

In this section we apply the ABC methodology to estimate the parameters of a COGARCH(1,1) model on real data. We use as in previous sections the series of Nasdaq 100 weekly returns from January 2000 to November 2012 which can be observed in figure 3.6.

Table 5.6 shows the obtained estimations for parameters, the standard deviations and the HPD intervals, using the ABC, HMC, data cloning and MCMC methods. In this case, we have used again 500000 simulations with a tolerance level of 0.01, and a sample of 5000 sets of parameters have been accepted as an obtained sample from the posterior distribution. We have undertaken parallel computing in codes, by means of Analytics and Weston (2014) with a significant diminishing computing times in a multicore machine.

Although all methods provide very similar estimates with small standard devi-

Estimations using ABC algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	9.573	0.406	$(8.919, 10.184)$
η	0.076	0.027	$(0.051, 0.133)$
φ	0.032	0.013	$(0.005, 0.048)$
Estimations using Hamiltonian Monte Carlo algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	9.637	2.064	$(6.155, 14.289)$
η	0.053	0.004	$(0.045, 0.062)$
φ	0.046	0.002	$(0.038, 0.049)$
Estimations using Data Cloning algorithm			
Parameter	Estimated Value	S.D.	95% Confidence intervals
β	9.041	2.890	$(5.457; 12.626)$
η	0.055	0.00042	$(0.050; 0.061)$
φ	0.05	0.00015	$(0.048; 0.052)$
Estimations using MCMC algorithm			
Parameter	Estimated Value	S.D.	HPD 0.95
β	9.661	2.113	$(6.140; 14.414)$
η	0.054	0.0043	$(0.045; 0.063)$
φ	0.047	0.0028	$(0.039; 0.050)$

Table 5.6: ABC estimations for COGARCH(1,1) model for Nasdaq daily returns data

ations, the ABC approach renders the smaller standard deviations for parameter β, but the higher for parameter η. Increasing the number of simulations these standard deviations could be reduced, with the consequent computational cost.

Conclusiones

En esta tesis se plantea la estimación de los parámetros de los modelos autorregresivos de heteroscedasticidad condicional generalizados, en tiempos discretos y continuos, mediante varias metodologías bayesianas y frecuentistas aproximadas. En concreto, se tratan los modelos GARCH y su extensión a tiempo continuo (COGARCH), siguiendo el planteamiento original de Klüppelberg et al. (2004). De este modo, se pueden modelizar series temporales observadas en intervalos irregulares de tiempo (véase Maller et al. (2008)).

Se presentan los diferentes métodos y se analizan sus propiedades matemáticas. Posteriormente, se efectúa un estudio de simulación para analizar la precisión de los estimadores y, finalmente, se muestran los resultados obtenidos en el análisis de series reales procedentes del índice bursátil Nasdaq entre enero del año 2000 y noviembre del año 2012.

Se presentan a continuación las siguientes observaciones y conclusiones generales:

- Se propone la aplicación de una metodología estándar MCMC en el modelo COGARCH(1,1) propuesto por Klüppelberg et al. (2004), considerando que los tiempos entre observaciones no son equidistantes (véase Maller et al. (2008)). Tras el estudio de sus propiedades y, a la vista de las pruebas de simulación, se concluye que el método proporciona estimaciones ajustadas de los parámetros del modelo, usando una única serie temporal de datos simulados.

  Se estudia también el caso de más de una trayectoria simulada del mismo modelo, observándose que los resultados son similares a los obtenidos par-

tiendo de una única trayectoria, por lo que se concluye que el uso de varias trayectorias de datos no mejora la estimación de los parámetros del modelo.

- Se plantea la estimación de los parámetros de los modelos GARCH y COGARCH, mediante la metodología de *Data Cloning* basada en el uso de réplicas de la serie de datos original (clones). Esta metodología permite obtener una aproximación numérica de los estimadores de máxima verosimilitud, mediante el uso instrumental de un algoritmo MCMC. Tanto en el caso de datos simulados, como en el caso de datos reales, las estimaciones de los parámetros son semejantes a las obtenidas mediante la metodología bayesiana estándar. Se puede destacar que el enfoque de Data Cloning permite usar cualquier distribución a priori, de manera que los resultados finales no son afectados por dicha elección (ver Lele et al. (2007)).

- Se estudia y presenta también la metodología *Hamiltonian Monte Carlo* (HMC) en los modelos GARCH y COGARCH. El método HMC es una variante de los métodos MCMC en donde se complementa el vector de parámetros a analizar, mediante una serie de variables auxiliares que permiten mejorar el rendimiento y la velocidad de convergencia del algoritmo. En el caso de los modelos GARCH y COGARCH se obtienen estimaciones análogas a las obtenidas usando un MCMC estándar, pero se reducen considerablemente los tiempos de computación en ambos modelos. Esta ventaja resulta especialmente relevante en el caso de la estimación de los parámetros del modelo COGARCH, donde los tiempos de estimación son habitualmente elevados.

- Finalmente, se aplica la metododología *Aproximate Bayesian Computation* (ABC) en los modelos GARCH y COGARCH. Los métodos ABC son de especial interés a la hora de estimar los parámetros del modelo COGARCH,

ya que no se requiere el uso de la función de verosimilitud. Esta función se aproxima habitualmente en los modelos GARCH y COGARCH, usando una pseudo-verosimilitud basada en la asunción de normalidad, y los estimadores frecuentistas que se obtienen a menudo son inestables. Se concluye que el estimador ABC propuesto resulta más eficiente que los obtenidos por otras metodologías, especialmente en el caso de la estimación del parámetro β.

La desventaja, en la práctica, de este algoritmo es el prolongado tiempo de computación que puede llegar a requerir. No obstante, se observa que las estimaciones obtenidas haciendo uso de un número razonable de iteraciones, no distan sustancialmente de las obtenidas utilizando una cantidad significativamente superior de las mismas.

Se concluye, en definitiva, que las metodologías bayesianas aplicadas y la metodología Data Cloning ofrecen soluciones fiables y eficientes para la estimación de los parámetros de los modelos GARCH y COGARCH, basadas en el rigor de sus planteamientos y la eficiencia de los algoritmos.

Appendix A

A.1 Simulation code for COGARCH(1,1) series

```
N <- 100
lam <- 1

# Simulation of time increments
library (e1071)
At <- rdiscrete (N, c(396,7,99,6), c(1,2,3,4))
B <- numeric(N) # Brownian component of Levy Process
pois <- numeric(N) # Poisson process
ele <- numeric(N) # Levy process

# Simulation of the Browian process
for (i in 2: N){
B[i] <- B[i-1] + rnorm(1,0,1) * sqrt(At)}

# Simulation of Poisson process
for (i in 1:N){
   Nt <- rpois(1,lam)
   for (j in 1:Nt){
   pois[i] <- pois[i] + rnorm(1,0,1)}}

# Simulation of Levy process
ele <- pois + B

# COGARCH parameters
bet <- 0.2
eta <- 0.25
fi <- 0.01

#Simulation of COGARCH process
G <- numeric(N) # GARCH increments
S2 <- numeric(N) # Variance of simulated GOGARCH
S2[1] <- bet/(eta-fi) # Volatility of simulated GARCH
G[1] <- 50 # Start value for series
for (i in 2:N) {
   G[i] <- G[i-1] + sqrt(S2[i-1])*(ele[i]-ele[i-1])
   S2[i] <- max(S2[i-1] + (bet - eta * S2[i-1])*At[i]
      + fi * S2[i-1]*(pow(ele[i]-ele[i-1],2), bet/eta)}

# Calculating increments
ytrue <- numeric(N)
for (i in 2:N){
   ytrue[i] <- G[i] - G[i-1]}

# Saving data in a file
write(ytrue, file="DatosSimulados.txt", append=FALSE, ncolumns=1)
write(At, file="tiempos.txt", append=FALSE, ncolumns=1)
write(G, file="COGARCH_simulado.txt", append=FALSE, ncolumns=1)
```

A.2 MCMC estimation code for COGARCH (1,1) parameters

```
# COGARCH model specification
modelo <- "
var
ro2[N], sigma2[N], P[N];

model {
   # Priors
   beta ~ dunif(0,100000)
   eta ~ dunif(fi,100000)
   fi ~ dunif(0,0.05)

   # Start values for variables
   sigma2_0 <- beta/(eta-fi)
   inct0 <- 1
   sigma2[1] <- max((beta*inct0) + (exp(-eta*inct0)
      *(sigma2_0 + fi*pow(yb[1],2))), beta/eta)
   ro2[1] <- ((sigma2_0 - (beta/(eta-fi)))*((exp((eta-fi)
      *inct[1])-1)/(eta-fi))) + ((beta*inct[1])/(eta-fi))
   P[1] <- 1/ro2[1]
   y[1] ~ dnorm(0, P[1])

   for(i in 2:N){
      y[i] ~ dnorm(0, P[i])
      sigma2[i] <- max((beta*inct[i]) + (exp(-eta*inct[i])
         *(sigma2[i-1] + (fi*pow(yb[i],2)))), beta/eta)
      ro2[i] <- ((sigma2[i-1] - (beta/(eta-fi)))*((exp((eta-fi)
         *inct[i])-1)/(eta-fi)))+((beta*inct[i])/(eta-fi))
      P[i] <- 1/ro2[i] }
}"

# Reading series and time intervals
ytrue <- scan(file="DatosSimulados.txt")
inct <- scan(file="tiempos.txt")

N <- length(G)
y <- ytrue
yb <- y
datos <- list("y"=y, "yb"=yb, "N"=N, "inct"=inct)
parametros <- c("fi", "beta", "eta")

library(runjags)
```

```
# Initial values
inibeta1 <- runif(1,0,10)
inifi1 <- runif(1,0,0.05)
inieta1 <- runif(1,inifi1,10)
inibeta2 <- runif(1,0,10)
inifi2 <- runif(1,0,0.05)
inieta2 <- runif(1,inifi2,10)
inits1 <- dump.format(list(fi=inifi1,eta=inieta1,beta=inibeta1))
inits2 <- dump.format(list(fi=inifi2,eta=inieta2,beta=inibeta2))

rutajags <- Sys.getenv("sistemjags")

# MCMC algorithm
resultados <- run.jags(model=modelo, inits=c(inits1,inits2),
   monitor=parametros, data=datos,jags=rutajags,
   n.chains=2, burnin=50000, sample=100000)
```

A.3 Simulation code for multiple trajectories of COGARCH(1,1)

```
N <- 100 # Number of time points for process
lam <- 1 # Average for poisson process
repes <- 50 # Number of trajectories
GTo <- NULL
YTo <- matrix(0,repes,N)

library(e1071)
At = rdiscrete(N, c(396,7,99,6), c(1,2,3,4))

# Model parameters:
bet <- 0.25
fi <- 0.02
eta <- 0.35

for (j in 1:repes){
   ele = numeric(N)
   # Levy process
   for (i in 1:length(ele)){
      Nt = rpois(1,lam)
      for (j in 1:Nt){
         ele[i] = ele[i] + rnorm(1,0,1)}}

   G <- numeric(N) # Simulated sample path
   S2 <- numeric(N) # Simulated sample path variance
   S2[1] <- bet /(eta - fi) # First value for variance
   G[1] <- 50 # First process value
   for (i in 2:N) {
      G[i] <- G[i-1] + sqrt(S2[i-1])*(ele[i]-ele[i-1])
      S2[i] <- max(S2[i-1] + (bet - eta*S2[i-1])*At[i] +
         fi*S2[i-1]*pow(ele[i]-ele[i-1],2), bet/eta)
      GTo <- rbind(GTo,G)}}

# Calculating the process increments
for (k in 1:repes){
   for (l in 2:N){
      YTo[k,l] <- GTo[k,l] - GTo[k,l-1]}}

data <- as.data.frame(YTo)

# Saving values and times
write.table(data, file="DatosSimuladosTO.txt", row.names=FALSE,
   col.names=FALSE, quote=FALSE)
write (At, file="tiemposTO.txt", append=FALSE, ncolumns=1)
```

A.4 MCMC estimation code for COGARCH(1,1) parameters using multiple trajectories

```
# COGARCH model specification
modelo <- "
var
ro2[m,N], sigma2[m,N], P[m,N];

model {

# Prior distributions and initial values
for (k in 1:m) {
   beta[k] ~ dunif(0,10000)
   fi[k] ~ dunif(0,0.05)
   eta[k] ~ dunif(fi[k],10000)
   sigma2_0[k] <- beta[k]/(eta[k]-fi[k])
   inct0[k] <- 1}

for (j in 1:m){
   sigma2[j,1] <- (beta[j] * inct0[j]) + (exp(-eta[j]
      * inct0[j]) * (sigma2_0[j] + fi[j] * pow(yb[j,1],2)))
   ro2[j,1] <- ((sigma2_0[j] - (beta[j]/(eta[j]-fi[j])))
      *((exp((eta[j]-fi[j])*inct[1])-1)/(eta[j]-fi[j])))
      + ((beta[j]*inct[1])/(eta[j]-fi[j]))
   P[j,1] <- 1/ro2[j,1]
   y[j,1] dnorm(0, P[j,1])
   for(i in 2:N){
      y[j,i] dnorm(0, P[j,i])
      sigma2[j,i] <- (beta[j] * inct[i]) + (exp(-eta[j]
         * inct[i]) * (sigma2[j,i-1] + (fi[j] * pow(yb[j,i],2))))
      ro2[j,i] <- ((sigma2[j,i-1] - (beta[j]/(eta[j]-fi[j])))
         * ((exp((eta[j]-fi[j])*inct[i])-1)/(eta[j]-fi[j])))
         + ((beta[j] * inct[i])/(eta[j] - fi[j]))
      P[j,i] <- 1/ro2[j,i]}}}"

# Reading data
ytrue <- read.table("DatosSimuladosTo.txt", header=FALSE)
inct <- scan(file="tiemposTO.txt")

N <- dim(ytrue)[2]
m <- dim(ytrue)[1]
ytrue <- as.matrix(ytrue)
y <- ytrue
yb <- y
datos <- list("y"=y, "yb"=yb, "N"=N, "m"=m, "inct"=inct)
parametros <- c("fi", "beta", "eta")
```

```
library(runjags)

# Initial values for parameters
inibeta <- runif(m,0,100)
inifi <- runif(m,0,0.05)
inieta <- runif(m,0.06,100)
iniciales <- dump.format(list(fi=inifi, eta=inieta,
beta=inibeta))

#MCMC algorithm
resultados <- run.jags(model=modelo, inits=iniciales,
monitor=parametros, data=datos,
jags="C:/JAGS/JAGS-3.2.0/i386/bin/jags-terminal.exe,"
n.chains=1, burnin=5000, sample=10000)

# Results
mean(resultados$summary$statistics[1:m,1]) # fi mean
mean(resultados$summary$statistics[(m+1):(2*m),1]) # beta mean
mean(resultados$summary$statistics[((2*m)+1):(3*m),1])# eta mean
mean(resultados$summary$quantiles[1:m,1])# fi 2.5%
mean(resultados$summary$quantiles[1:m,5]) # fi 97.5%
mean(resultados$summary$quantiles[(m+1):(2*m),1]) # beta 2.5%
mean(resultados$summary$quantiles[(m+1):(2*m),5]) # beta 97.5%
mean(resultados$summary$quantiles[((2*m)+1):(3*m),1]) # eta 2.5%
mean(resultados$summary$quantiles[((2*m)+1):(3*m),5]) # eta 97.5%
```

A.5 Data cloning estimation code for GARCH(1,1) parameters

```
library(dclone)

# Jags code
garch.function <- function() {

# Prior distributions
alpha0 ~ dunif(0,100)
alpha1 ~ dlnorm(0,3.562145)
bet ~ dunif(0,1 - alpha1)

for (j in 1:k){
   h[1,j] <- 1
   P[1,j] <- 1/h[1,j]
   for (i in 2:n) {
      y[i,j] dnorm(0, P[i,j])
      h[i,j] <- alpha0 + alpha1 * pow(y[i-1,j],2)
         + bet * h[i-1,j]
      P[i,j] <- 1/h[i,j]}}
}

# Reading data from the file
ytrue <- scan(file="Grachsimulado.txt")
y <- ytrue
n <- length(ytrue)

# Creating dclone format
bhdat <- list(y=dcdim(data.matrix(y)), n=n, k=1)
dcbhdat <- dclone(bhdat, n.clones=50, multiply="k", unchanged="n")

# Initial values for parameters
inialpha0 <- runif(1,0,100)
inialpha1 <- runif(1,0,3.562145)
inibeta <- runif(1,0,1)
iniciales <- list(alpha0=inialpha0, alpha1=inialpha1, bet=inibeta)

# function to update initials
ifun <- function(model, n.clones) {
   list(alpha0=coef(model)[1],
   alpha1=coef(model)[2],
   bet=coef(model)[3])}
```

```
# Data cloning algorithm
jmod <- dc.fit(bhdat, c("alpha0", "alpha1", "bet"),
   garch.function, inits=iniciales,n.clones=c(50),
   multiply="k", unchanged="n", initsfun=ifun)

# Results summary with DC and SE
summary(jmod)
dct <- dctable(jmod)
confint(jmod)
```

A.6 Data cloning estimation code for COGARCH(1,1) parameters

```
library(dclone)

cogarch.estimator <- function() {

   # Prior distributions and initial values
   beta ~ dunif(priors[1,1],priors[1,2])
   eta ~ dunif(priors[2,1],priors[2,2])
   fi ~ dunif(priors[3,1],priors[3,2])
   sigma2_0 <- beta /(eta-fi)
   inct0 <- 1

   # Clones
   for (j in 1:k){
      sigma2[1,j] <- (beta*inct0) + (exp(-eta*inct0)
         *(sigma2_0 + fi * pow(yb[1,j],2)))
      ro2[1,j] <- ((sigma2_0 - (beta/(eta-fi)))
         *((exp((eta-fi)*inct[1,j])-1)/(eta-fi)))
         + ((beta*inct[1,j])/(eta-fi))
      P[1,j] <- 1/ro2[1,j]
      y[1,j] ~ dnorm(0, P[1,j])
      for(i in 2:n){
         y[i,j] ~ dnorm(0, P[i,j])
         sigma2[i,j] <- (beta*inct[i,j])
            + (exp(-eta*inct[i,j]) * (sigma2[i-1,j]
            + (fi*pow(yb[i,j],2))))
         ro2[i,j] <- ((sigma2[i-1,j] - (beta/(eta-fi)))
            *((exp((eta-fi)*inct[i,j])-1)/(eta-fi)))
            + ((beta*inct[i,j])/(eta - fi))
         P[i,j] <- 1/ro2[i,j] }}
}

# Reading data from the file
ytrue <- scan(file="DatosSimulados.txt")
inct <- scan(file="tiempos.txt")

y <- ytrue
yb <- y
n <- length(y)

# Initial values for the parameters
inibeta <- runif(1,0,1)
inifi <- runif(1,0,0.1)
inieta <- runif(1,inifi,1)
iniciales <- list(fi=inifi, eta=inieta, beta=inibeta)
```

```
upfun <- function(x) {
if (missing(x)) {
   return(cbind(c(0,0.0501,0), c(100000,100000,0.05)))}
else{
   par <- coef(x)}}

updat <- list(y=dcdim(data.matrix(y)), yb=dcdim(data.matrix(yb)),
inct=dcdim(data.matrix(inct)), n=n, k=1, priors=upfun())

# Data cloning algorithm
jjmod <- dc.fit(updat, c("fi", "beta", "eta"), cogarch.estimador,
   inits=iniciales, n.clones = 20, multiply="k", unchanged="n",
   update="priors", updatefun=upfun)

# Results summary
summary(jjmod)
confint(jjmod)
vcov(jjmod)
```

A.7 Hamiltonian Monte Carlo estimation code for GARCH(1,1) parameters

```
library(rstan)

programa <- '
   data {
      int<lower=0> T;
      real r[T];
      real<lower=0> sigma1; }

   parameters {
      real<lower=0> alpha1;
      real<lower=0,upper=1> alpha2;
      real<lower=0, upper=(1-alpha2)> bet; }

   transformed parameters {
      real<lower=0> sigma[T];
      sigma[1] <- sigma1;
      for (t in 2:T)
         sigma[t] <- sqrt(alpha1 + alpha2*pow(r[t-1], 2)
            + bet*pow(sigma[t-1],2));}

   model {r ~ normal(0,sigma);}'

# Reading data from the file
datossimulados <- scan(file="Grachsimulado.txt")
n <- length(datossimulados)
datos <- list(T=n, r=datossimulados, sigma1=1)

#HMC algorithm
resultados <- stan(model code=programa, data=datos, iter=5000,
                   chains=3)

# Summarizing results
d <- as.data.frame(resultados)
parametros <- d[,1:3]
apply(parametros,2,mean)
apply(parametros,2,sd)
apply(paras,2,quantile, probs = c(0.05, 0.5, 0.975), na.rm=TRUE)
```

A.8 Hamiltonian Monte Carlo estimation code for COGARCH(1,1) parameters

```
library(rstan)

programa <- '
   data {
      int<lower=0> N;
      real y[N];
      real yb[N];
      real inct[N];
      int<lower=0> inct 0; }

   parameters {
      real<lower=0> beta;
      real<lower=0,upper=0.05> fi;
      real<lower=fi,upper=10> eta; }

   transformed parameters {
      real<lower=0> sigma2[N];
      real<lower=0> ro2[N];
      real<lower=0> sigma2 0;
      sigma2 0 <- beta/(eta-fi);
      sigma2[1] <- (beta*inct 0) + (exp(-eta*inct 0)
         *(sigma2 0 + fi*pow(yb[1],2)));
      ro2[1] <- sqrt(((sigma2 0 - (beta/(eta-fi)))
         *((exp((eta-fi)*inct[1])-1)/(eta-fi)))
         + ((beta*inct[1])/(eta-fi)));

      for(i in 2:N){
      sigma2[i] <- (beta*inct[i])
         + (exp(-eta*inct[i])
         * (sigma2[i-1] + (fi*pow(yb[i],2))));
      ro2[i] <- sqrt(((sigma2[i-1]
         - (beta/(eta-fi)))
         *((exp((eta-fi)*inct[i])-1)/(eta-fi)))
         + ((beta*inct[i])/(eta - fi)));}}

   model {y ~ normal(0, ro2); }'

# Reading data from the file
ytrue <- scan(file="DatosSimulados.txt")
inct <- scan(file="tiempos.txt")
```

```
y <- ytrue
yb <- y
N <- length(y)
inct <- inct[-1]
datos <- list(y=y, yb=yb, N=N, inct=inct, inct 0=1)

# HMC algorithm
resultados <- stan(model code=programa, data=datos, iter=5000,
                   chains=3)

# summarizing results
print(resultados, pars=c("beta","fi","eta"), digits=5)
```

A.9 ABC estimation code for GARCH(1,1) parameters

```
library(e1071)
library(fdth)
library(fGarch)
library(EasyABC)

# Function to simulate the GARCH(1,1) process

simuGARCH <- function(alfa1, alfa2, bet, calor=50, N=nsamples){
   # Warm up values
   cuantos <- (calor+N)
   y <- numeric(cuantos)  # GARCH increments
   S2 <- numeric(cuantos) # Volatility of simulated GARCH
   S2[1] <- 1
   nu <- rnorm(cuantos,0,1)

   # Condition "alfa2 + beta < 1"
   if ((alfa2+bet) > 1){return(NA)}

   for (i in 2:cuantos){
      # Volatility
      S2[i] <- alfa1 + (alfa2*(y[i-1]^2)) + (bet*S2[i-1])
      # Returns
      y[i] <- sqrt(S2[i])*nu[i]
   }
   sale <- y[(calor+1):cuantos]
   return(sale)
 }

# Reading values from the file
datos <- scan(file="Grachsimulado.txt")
nsamples <- length(datos)

# Maximum likelihood estimations using fGarch library
estima <- garchFit(~garch(1,1), data=datos,
          include.mean=FALSE, trace=FALSE)

alfa1hat <- estima@fit$coef[[1]]
alfa2hat <- estima@fit$coef[[2]]
betahat  <- estima@fit$coef[[3]]

m <- c(alfa1hat, alfa2hat, betahat)

cat("alpha1= ", m[1], "\n")
cat("alpha2= ", m[2], "\n")
cat("beta= ", m[3], "\n")

# Bounds for prior distributions based on QMLE
```

```
ini1.1 <- max((m[1]-0.2),1e-10)
ini1.2 <- min((m[1]+0.2),100)

ini2.1 <- max((m[2]-0.2),1e-10)
ini2.2 <- min((m[2]+0.2),3.562145)

ini3.1 <- max((m[3]-0.2),1e-10)
ini3.2 <- min((m[3]+0.2),1)

# Number of simulations
itera <- 500000

# Prior definition
alfa1_prior <- runif(itera, ini1.1, ini1.2)
alfa2_prior <- runif(itera, ini2.1, ini2.2)
beta_prior <- runif(itera, ini3.1, ini3.2)

param_prior <- cbind(alfa1_prior, alfa2_prior, beta_prior)
sumstat <- matrix(NA, itera, 3)

for (i in 1:itera){
   y.sim <- simuGARCH(alfa1=param_prior[i,1], alfa2=param_prior[i,2],
                      bet=param_prior[i,3], N=nsamples)

   sumstat[i,] <- c(mean(y.sim, na.rm=TRUE),
                    var(y.sim, na.rm=TRUE), mad(y.sim, na.rm=TRUE))
}

# Eliminate the NA values obtained with the restrictions
sumstat <- sumstat[complete.cases(sumstat),]

# Statistics to compare in the ABC algorithm
summarydata <- c(mean(datos,na.rm=TRUE),
                 var(datos,na.rm=TRUE), mad(datos, na.rm=TRUE))

colnames(sumstat) <- c("mean", "var", "mad")

# ABC algorithm by using the EasyABC library
rej <- abc(target=summarydata, param=param_prior,
           sumstat, tol=0.005, method="rejection")

# Showing the results
val.alpha1 <- rej$unadj.values[,1]
val.alpha2 <- rej$unadj.values[,2]
val.beta <- rej$unadj.values[,3]

med.alpha1 <- mean(val.alpha1)
med.alpha2 <- mean(val.alpha2)
med.beta <- mean(val.beta)

cat("Media Parametro alpha1: ", med.alpha1, "\n")
cat("Media Parametro alpha2: ", med.alpha2, "\n")
cat("Media Parametro beta: ", med.beta, "\n")
```

```
sd(val.alpha1)
sd(val.alpha2)
sd(val.beta)

cat("Parametro alpha1", "\n")
quantile(val.alpha1, c(0.025,0.5,0.975))
cat("Parametro alpha2", "\n")
quantile(val.alpha2, c(0.025,0.5,0.975))
cat("Parametro beta", "\n")
quantile(val.beta, c(0.025,0.5,0.975))
```

A.10 ABC-MCMC estimation code for GARCH(1,1) parameters

```
library(EasyABC)
library(e1071)
library(fdth)
library(fGarch)

# Function to simulate the GARCH(1,1) process
simuGARCH <- function(alfa1, alfa2, bet, calor=50, N=nsamples){
   # Warm up values
   cuantos <- (calor+N)
   y <- numeric(cuantos)  # GARCH increments
   S2 <- numeric(cuantos) # Volatility of simulated GARCH
   S2[1] <- 1
   nu <- rnorm(cuantos,0,1)

   # Condition alfa2 + beta < 1
   if ((alfa2+bet) > 1){bet <- (1-alfa2)}

   for (i in 2:cuantos){
      # Volatility
      S2[i] <- alfa1 + (alfa2*(y[i-1]^2)) + (bet*S2[i-1])
      # Returns
      y[i] <- sqrt(S2[i])*nu[i]
   }
   sale <- y[(calor+1):cuantos]
   return(sale)
}

# Reading values from the file
datos <- scan(file="Grachsimulado.txt")
nsamples <- length(datos)

# Maximum likelihood estimations using fGarch library
estima <- garchFit(~garch(1,1), data=datos,
          include.mean=FALSE, trace=FALSE)

summarydata <- c(mean(datos,na.rm=TRUE), var(datos,na.rm=TRUE),
                 mad(datos, na.rm=TRUE))

modelo <- function(para){
   # parameters: alpha1, alpha2, beta
   samples <- simuGARCH(alfa1=para[1], alfa2=para[2],
   bet=para[3], N=nsamples)
   estima.simu <- garchFit(~garch(1,1), data=samples,
   include.mean=FALSE, trace=FALSE)
   ay1 <- estima.simu@fit$coef[[1]]
   ay2 <- estima.simu@fit$coef[[2]]
   ay3 <- estima.simu@fit$coef[[3]]
   return(c(ay1, ay2, ay3))
```

```
}

# Bounds for prior distributions based on QMLE
ini1.1 <- max((estima@fit$coef[[1]]-0.2),1e-10)
ini1.2 <- min((estima@fit$coef[[1]]+0.2),100)

ini2.1 <- max((estima@fit$coef[[2]]-0.1),1e-10)
ini2.2 <- min((estima@fit$coef[[2]]+0.1),3.562145)

ini3.1 <- max((estima@fit$coef[[3]]-0.1),1e-10)
ini3.2 <- min((estima@fit$coef[[3]]+0.1),1)

priordist <- list(c("unif",ini1.1,ini1.2),
   c("unif",ini2.1,ini2.2), c("unif",ini3.1,ini3.2))

# Number of simulations
n_simu <- 500000

# MCMC-ABC algorithm based on Marjoram method by using the EasyABC
# library

Mar_ori <- ABC_mcmc(method="Marjoram", model=modelo, prior=priordist,
summary_stat_target=summarydata, n_rec=n_simu, prior_test="X2+X3 <1")

#Showing the results

val.alpha1 <- Mar_ori$param[(n_simu/2):n_simu,1]
val.alpha2 <- Mar_ori$param[(n_simu/2):n_simu,2]
val.beta <- Mar_ori$param[(n_simu/2):n_simu,3]

X11()
hist(val.alpha1, main=expression(paste("Posterior for ",alpha[1])),
xlab=expression(alpha[1]))
X11()
hist(val.alpha2, main=expression(paste("Posterior for ",alpha[2])),
xlab=expression(alpha[2]))
X11()
hist(val.beta, main=expression(paste("Posterior for ",beta)),
xlab=expression(beta))

med.alpha1 <- mean(val.alpha1)
med.alpha2 <- mean(val.alpha2)
med.beta <- mean(val.beta)

cat("Media Parametro alpha1: ", med.alpha1, "\n")
cat("Media Parametro alpha2: ", med.alpha2, "\n")
cat("Media Parametro beta: ", med.beta, "\n")

sd(val.alpha1)
sd(val.alpha2)
sd(val.beta)

cat("Parametro alpha1", "\n")
```

```
quantile(val.alpha1, c(0.025,0.5,0.975))
cat("Parametro alpha2", "\n")
quantile(val.alpha2, c(0.025,0.5,0.975))
cat("Parametro beta", "\n")
quantile(val.beta, c(0.025,0.5,0.975))
```

A.11 ABC estimation code for COGARCH(1,1) parameters

```
library(COGARCH)
library (e1071)
library(EasyABC)

# Reading values from the file

G <- scan(file="COGARCH_simulado.txt")
inct <- scan(file="tiempos.txt")
nG <- length(G)
y <- diff(G)

cuando <- cumsum(inct)

# Function to calculate the minimum volatility and its range
min_vol <- function (r) {
   nr <- length(r)
   vol <- numeric(nr)
   for (i in (1:nr)){
      vol[i] <- (r[i]-mean(r))^2}
   return (c(min(vol), max(vol)-min(vol)))
}

# Function to calculate COV(r_t^2,r_{t-1}^2)
cov_corr_sq <- function (r) {
   nr <- length(r)
   r1 <- numeric(nr-1)
   for (i in (2:nr)){
      r1[i-1] <-r[i]}
      r <- r[1:(nr-1)]
      rsqrt=(r^2)
      r1sqrt=(r1^2)
      return(c(cov(rsqrt,r1sqrt),cor(rsqrt,r1sqrt)))
}


# Statistics to compare in the ABC algorithm
summarydata <- c(min_vol(G)[1], min_vol(G)[2],  cov_corr_sq(G)[2],
                 mad(G, na.rm=TRUE),var(G, na.rm=TRUE))

# Power moment method estimations
momento <- MMestimation(y, d=26, explicit=TRUE)

betahat <- momento$theta@beta
etahat <- momento$theta@eta
phihat <- momento$theta@phi

if(betahat < 1e-06) betahat <- 1.5e-06
if(etahat < 1e-02) etahat <- 1.5e-02
```

```
if(phihat < 1e-02) phihat <- 1.5e-02
if((etahat-phihat) < 1e-02){etahat <- phihat + 0.01 }

m <- c(etahat, betahat, phihat)

# Bounds for the prior distributions based
# on the power moment estimations

ini1.1 <- max((m[1]-0.5), min(m[3]+0.50001,0.05001))
ini1.2 <- min((m[1]+0.5),100)

ini2.1 <- max((m[2]-0.5),1e-10)
ini2.2 <- min((m[2]+0.5),100)

ini3.1 <- max((m[3]-0.5),1e-10)
ini3.2 <- min((m[3]+0.5),0.05)

# Number of calculated simulations
itera <- 500000

# Prior distributions
eta_prior <- runif(itera, ini1.1, ini1.2)
beta_prior <- runif(itera, ini2.1, ini2.2)
phi_prior <- runif(itera, ini3.1, ini3.2)

param_prior <- cbind(eta_prior, beta_prior, phi_prior)
sumstat <- matrix(NA, itera, 5)

# Estimating the series from the parameters and
# calculating the statistics for the series

for (i in 1:itera){
   prm.sim <- COGprm(eta=param_prior[i,1],
      beta=param_prior[i,2], phi=param_prior[i,3])
   prv.sim <- rCOGARCH(prm.sim, rnorm,
   obstimes=cuando, sigmaSq0=1, G0=0, lambda=1)
   y.sim <- diff(prv.sim@G[,1])
   sumstat[i,] <- c(min_vol(prv.sim@G[,1])[1],
   min_vol(prv.sim@G[,1])[2], cov_corr_sq(prv.sim@G[,1])[2],
   mad(prv.sim@G[,1], na.rm=TRUE), var(prv.sim@G[,1], na.rm=TRUE))
}

colnames(sumstat) <- c("min_vol","rango_vol","cor","mad","var")

#ABC rejection method

rej <- abc(target=summarydata, param=param_prior,
           sumstat, tol=0.01, method="rejection")

# Parameters estimations
(eta_posmean_abc <- mean(rej$unadj.values[,1]))
(beta_posmean_abc <- mean(rej$unadj.values[,2]))
(phi_posmean_abc <- mean(rej$unadj.values[,3]))
```

```
#HPD eta
quantile(rej$unadj.values[,1], c(0.05,0.5,0.95))
#HPD beta
quantile(rej$unadj.values[,2], c(0.05,0.5,0.95))
#HPD phi
quantile(rej$unadj.values[,3], c(0.05,0.5,0.95))

# Standard deviations
(eta_sd <- sd(rej$unadj.values[,1]))
(beta_sd <- sd(rej$unadj.values[,2]))
(phi_sd <- sd(rej$unadj.values[,3]))
```

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on 19*(6), 716–723.

Alder, B. J. and T. Wainwright (1959). Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics 31*(2), 459–466.

Analytics, R. and S. Weston (2014). *doParallel: Foreach parallel adaptor for the parallel package*. R package version 1.0.8.

Bibbona, E., I. Negri, A. Arcagni, and L. Mercuri (2014). *COGARCH: Simulation and inferenc for COGARCH(1,1) processes*. R package version 1.3-1/r32.

Billingsley, P. (2008). *Probability and measure*. John Wiley & Sons.

Black, F. (1976). Studies of stock price volatility changes. *Proceedings from the American Statistical Association, Business and Economic Statistics Section*, 177–181.

Black, F. and M. Scholes (1973). The pricing of options and corporate liabilities. *The journal of political economy*, 637–654.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics 31*(3), 307–327.

Bollerslev, T. (2008). Glossary to ARCH (GARCH). *Russell The Journal Of The Bertrand Russell Archives 49*.

Box, G. E., G. M. Jenkins, and G. C. Reinsel (1970). *Time series analysis: forecasting and control*. Holden-Day.

Breusch, T. (1978). Testing for autocorrelation in dynamic linear models*. *Australian Economic Papers 17*(31), 334–355.

Brooks, S. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics 7*(4), 434–455.

Chan, K.-S. and B. Ripley (2012). *TSA: Time Series Analysis*. R package version 1.01.

Corradi, V. (2000). Reconsidering the continuous time limit of the garch (1, 1) process. *Journal of Econometrics 96*(1), 145–153.

Denwood, M. J. (2011). *runjags: Run Bayesian MCMC Models in the BUGS syntax from Within R*. R package version 0.9.9-2.

Dickey, D. and W. Fuller (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association 74*(366a), 427–431.

Drost, F. and T. Nijman (1993). Temporal aggregation of garch processes. *Econometrica 61*(4), 909–927.

Drost, F. and B. Werker (1996). Closing the garch gap: Continuous time garch modeling. *Journal of Econometrics 74*(1), 31–57.

Duan, J. (1997). Augmented garch (¡ i¿ p¡/i¿,¡ i¿ q¡/i¿) process and its diffusion limit. *Journal of Econometrics 79*(1), 97–127.

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). Hybrid Monte Carlo. *Physics letters B 195*(2), 216–222.

Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica 50*(4), 987–1008.

Francq, C. and J. Zakoian (2010). *GARCH models: structure, statistical inference and financial applications*. Wiley.

Francq, C., J.-M. Zakoian, et al. (2004). Maximum likelihood estimation of pure garch and arma-garch processes. *Bernoulli 10*(4), 605–637.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2014). *Bayesian data analysis*, Volume 2. Taylor & Francis.

Godfrey, L. (1978). Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica: Journal of the Econometric Society*, 1293–1301.

Haug, S., C. Klüppelberg, A. Lindner, and M. Zapp (2005). Estimating the COGARCH(1,1) model-a first go.

Hosking, J. R. (1981). Fractional differencing. *Biometrika 68*(1), 165–176.

Ishwaran, H. (1999). Applications of hybrid Monte Carlo to Bayesian generalized linear models: Quasicomplete separation and neural networks. *Journal of Computational and Graphical Statistics 8*(4), 779–799.

Jaakkola, T. S. and M. I. Jordan (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing 10*(1), 25–37.

Jabot, F., T. Faure, N. Dumoullin, and M. N. Dumoulin (2014). Package EasyABC.

Joyce, P. and P. Marjoram (2008). Approximately sufficient statistics and Bayesian computation. *Statistical applications in genetics and molecular biology 7*(1).

Kallsen, J. and M. Taqqu (1998). Option pricing in ARCH-type models. *Mathematical Finance 8*(1), 13–26.

Kazmerchuk, Y., A. Swishchuk, and J. Wu (2005). A continuous-time GARCH model for stochastic volatility with delay. *Canadian Applied Mathematics Quarterly 13*(2), 123–149.

Klüppelberg, C., A. Lindner, and R. Maller (2004). A continuous-time GARCH process driven by a Lévy process: Stationarity and second-order behaviour. *Journal of Applied Probability 41*(3), 601–622.

Klüppelberg, C., A. Lindner, and R. Maller (2006). Continuous time volatility modelling: COGARCH versus ornstein–uhlenbeck models. *From stochastic calculus to mathematical finance*, 393–419.

Lele, S., B. Dennis, and F. Lutscher (2007). Data cloning: easy maximum likelihood estimation for complex ecological models using bayesian Markov chain Monte Carlo methods. *Ecology Letters 10*(7), 551–563.

Lele, S., K. Nadeem, and B. Schmuland (2010). Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association 105*(492), 1617–1625.

Ljung, G. and G. Box (1978). On a measure of lack of fit in time series models. *Biometrika 65*(2), 297–303.

Lorenz, R. (2006). *Weak approximation of stochastic delay differential equations*

with bounded memory by discrete time series. Ph. D. thesis, Ph. D. thesis, Humbold University of Berlin, Berlin, Germany.

Maller, R. A., G. Müller, A. Szimayer, et al. (2008). GARCH modelling in continuous time for irregularly spaced time series data. *Bernoulli 14*(2), 519–542.

Mandelbrot, B. (1963). The variation of certain speculative prices. *The journal of business 36*(4), 394–419.

Marín, J. M., M. T. Rodríguez-Bernal, and E. Romero (2015). Data cloning estimation of garch and cogarch models. *Journal of Statistical Computation and Simulation 85*(9), 1818–1831.

Marjoram, P., J. Molitor, V. Plagnol, and S. Tavaré (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences 100*(26), 15324–15328.

McKinley, T., A. R. Cook, and R. Deardon (2009). Inference in epidemic models without likelihoods. *The International Journal of Biostatistics 5*(1).

Meddahi, N. and E. Renault (2004). Temporal aggregation of volatility models. *Journal of Econometrics 119*(2), 355–379.

Møller, J., A. N. Pettitt, R. Reeves, and K. K. Berthelsen (2006). An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika 93*(2), 451–458.

Müller, G. (2010). MCMC estimation of the COGARCH(1,1) model. *Journal of Financial Econometrics 8*(4), 481–510.

Nakatsuma, T. (1998). A markov-chain sampling algorithm for GARCH models. *Studies in Nonlinear Dynamics and Econometrics 3*(2), 107–117.

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

Neal, R. M. (1995). *Bayesian learning for neural networks*. Ph. D. thesis, University of Toronto.

Nelson, D. (1990). Arch models as diffusion approximations. *Journal of Econometrics 45*(1-2), 7–38.

Plummer, M. et al. (2003). Jags: A program for analysis of bayesian graphical models using gibbs sampling. *URL http://citeseer. ist. psu. edu/plummer03jags. html*.

Pritchard, J. K., M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman (1999). Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution 16*(12), 1791–1798.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Revuz, D. and M. Yor (1999). *Continuous martingales and Brownian motion*, Volume 293. Springer Science & Business Media.

Rubin, D. (1984). Bayesianly justifiable and relevant frequency calculations for the applies statistician. *The Annals of Statistics 12*(4), 1151–1172.

Schmidt, M. N. (2009). Function factorization using warped gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 921–928. ACM.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464.

Sólymos, P. (2010). dclone: Data cloning in R. *The R Journal 2*(2), 29–37.

Stan Development Team (2014a). Rstan: the r interface to stan, version 2.5.0.

Stan Development Team (2014b). *Stan Modeling Language Users Guide and Reference Manual, Version 2.5.0.*

Tankov, P. (2003). *Financial modelling with jump processes*. CRC press.

Tavare, S., D. Balding, R. Griffiths, and P. Donnelly (1997). Inferring coalescence times from dna sequence data. *Genetics 145*(2), 505–518.

Tierney, L. and J. B. Kadane (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the american statistical association 81*(393), 82–86.

Tsay, R. S. (2005). *Analysis of financial time series*, Volume 543. John Wiley & Sons.

Wang, Y. (2002). Asymptotic nonequivalence of garch models and diffusions. *The Annals of Statistics 30*(3), 754–783.

Wuertz, D., C. B. Miklovic, P. Chausse, S. RUnit, and M. Y. Chalabi (2013). Package fgarch. Technical report, working paper/manual, 09.11. 2009. URL http://cran. r-project. org/web/packages/fGarch/fGarch. pdf.