UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIENCIAS EXATAS
DEPARTAMENTO DE ESTATÍSTICA

BOOTSTRAP METHODS FOR GENERALIZED AUTOREGRESSIVE MOVING AVERAGE MODELS

Matheus de Vasconcellos Barroso

Belo Horizonte
2018

Matheus de Vasconcellos Barroso

BOOTSTRAP METHODS FOR GENERALIZED AUTOREGRESSIVE MOVING AVERAGE MODELS

Trabalho apresentada como requisito para obtenção do título de Mestre em Estatística pela Universidade Federal de Minas Gerais

Professora orientadora: Glaura da Conceição Franco

Belo Horizonte

2018

Folha de aprovação

Abstract

This final paper aims to find a suitable Bootstrap Method for the Generalized Autoregressive Moving Average Model. The focus is on the Moving Block Bootstrap (MBB) resampling scheme with its performance being evaluated through a Monte Carlo study and contrasted to their asymptotic Gaussian counterpart. It is stablished that the aforementioned resampling procedure can generate good estimates of parameters *bias* and *confidence intervals*. Though, the results rely heavily on the simulated model parameters and block lengths used in the MBB procedure.

Key-words: GARMA models, Time- Series BOOTSTRAP, MBB

FIGURES LIST

TABLES LIST

CONTENTS

1 INTRODUCTION

This monograph is concerned with the implementation and evaluation of bootstrap parameter bias (the difference between the parameter true value and its estimator expected value) and confidence interval estimation in the context of the so-called *Generalized Autoregressive Moving Average models*, henceforth *GARMA* models, proposed by Benjamin et al. (2003). The estimation performance is accessed through a Monte Carlo simulation scheme.

GARMA models can be easily viewed as an extension of the linear regression model, to accommodate both time-series and non-Gaussian observations. Taking into account the time dependence between dependent and explanatory variables, one of the most used procedures is the Gaussian *Autoregressive Moving Average* (ARMA) model (Box and Jenkins, 1976) and, with regard to the non-gaussian behavior**,** the Generalized Linear Model (GLM) (McCullagh and Nelder, 1989) is often employed. If we combine both previous procedures we arrive at the *GARMA* model, that is, the *GARMA* model is simply the application of an ARMA process to model the conditional mean (through an appropriate *link function*) of the dependent variable within the exponential family of distributions.

Parameter estimation and inference in the *GARMA* model are based on maximum likelihood asymptotic results, though, for small length series, those results might not hold. For this reason, we employ bootstrap methods to help us access the accuracy of our estimates. Traditional bootstrap methods rely on the independent observations assumption, and as such, additional care must be taken when choosing a suitable resampling scheme that incorporates its intrinsic temporal dependence. A wide range of bootstrap methods are available, however, the implementation in the *GARMA* context is not easy and their performance is not homogenous.

Owing to this, the performance of parameter bias and confidence interval estimation is evaluated in light of the Moving Block Bootstrap (MBB) (Kunsch, 1989) resampling scheme. That is, our main goal is to examine whether the MBB can generate good and consistent results, and as such, if it can be widely applied in the GARMA framework. Additionally, as the block length is a relevant input for this class of resampling scheme, the algorithms of Hall, Horowitz and Jing (1995) and Lahiri, Furukawa and Lee (2007) are implemented. Furthermore, a modified version of the former algorithm is put forth and some remarks regarding the optimal block length choice are made.

A Monte Carlo study with 1000 simulations for each model is used for parameter bias and confidence interval evaluation. In the former, we check whether the bootstrap bias corrected distribution is centered on its true value, whereas in the latter we compute the bootstrap

replications coverage rate (i.e. the count of intervals containing the parameter true value divided by the number of simulations) and contrast it to the asymptotic Gaussian (Normal) interval.

We study a wide variety of models from both the continuous and discrete cases, with simulations restricted to the *Gamma* and *Poisson GARMA* models, respectively.

The outline of this monograph is as follows: in Section 2 we provide some historical references for both the *GARMA* model and the bootstrap method in the time-series perspective. The third Section introduces the *GARMA* model, Section 4 the bootstrap schemes used in the study, Section 5 analyzes the simulation results, Section 6 a real data example and Section 7 gives some concluding remarks. The Appendix provides additional information suppressed in the text for the sake of concision.

2 LITERATURE REVIEW

This section is divided in two subsections; the first is about the *GARMA* model and the second the bootstrap method applied in the time series framework.

2.1 The GARMA model

The *GARMA* model was introduced and formalized in the paper of Benjamin et al. (2003); however, many ideas present in the model were introduced before in the time series literature (see for example Zeger and Qaqish (1988)). Similar results applied in the context of discrete time series can be found at Davis et al. (1999), where the Generalized Linear Autoregressive Model (GLARMA) is developed. For applications of the latter the reader can refer to Jung et al. (2006).

We will bear in mind Benjamin's et. al. (2003) notion of the *GARMA* model as an extension to the Gaussian ARMA model, which traces back to the work of Cox et al. (1981). In the text, the author claims that in the non-Gaussian series perspective:

> It would be desirable to have a general exponential family formulation. Such models could be formulated as 'observation driven', or as 'parameter driven', the latter being instances of latent structure models. (Cox et al., 1981 p.101)

If we make use of Cox et al. (1981) terminology, the *GARMA* model suits the class of observation driven models, as opposed to parameter driven (state-space) one.

Some of the advantages of the state-space approach are its flexibility and ability to model the behavior of different components of the series separately and then aggregate the submodels to form an overall model for the time series (Durbin and Koopman, 2000). On the other hand, the improved flexibility comes at the expense of complicate estimation process and crude approximations (Benjamin et al., 2003). The study of state-space models is beyond the scope of this monograph and the reader is encouraged to recur to Durbin and Koopman (2000) for further details.

The paper of Benjamin et al. (2003) extends the work of Zeger and Qaqish (1988) and Li (1994). The former implements a Quasi-Likelihood Markov model in the conditional moments, in the same sense of McCullagh and Nelder (1989) where the marginal moments are employed. Nevertheless, their focus is on autoregressive models, such as autoregressive conditionally heteroscedastic (ARCH) models for example. The latter emphasizes the moving average perspective into the GLM context; their formulation is general enough to accommodate both autoregressive and moving average processes, though no general formal treatment is provided in this case.

With respect to non-Gaussian autoregressive models, we can also highlight the work of Grunwald et al. (2000), in which the authors develop the class of conditional linear AR(1) models (CLAR(1) models). This is a first-order conditional linear autoregressive structure, that subsumes a wide variety of models previously proposed in the literature, regardless of their generating method: innovation, conditional distribution, random coefficient, thinning and random coefficient thinning. For instance, consider the model of Zeger and Qaqish (1988), which is included in the conditional distribution method. The authors state the assumptions under which the CLAR(1) stationary mean can be derived and also its stationary variance (given a quadratic variance function premise). They also provide the conditions for a stationary (ergodic) distribution.

A detailed analysis of the *GARMA* model can be found at the book of Kedem and Fokianos (2002), where the authors explain in details the model, estimation process, residual analysis, many applications to discrete process and an exposition of the main models in the literature.

Some further developments have been proposed regarding *GARMA* models. Woodard et al. (2011) have shown their strict stationarity (the distribution of the process does not depend on time) from two perspectives. In the first approach they postulate under which conditions *GARMA* models have a unique stationary distribution and in the second they show stationarity and ergodicity of a perturbed version of the model. Subsequently, they relate the original to the

perturbed processes and conclude that the latter has parameter estimates arbitrarily close to the former.

Afterwards, Briet et al. (2013) extended the *GARMA* class into the Generalized Seasonal Autoregressive Integrated Moving Average models (*GSARIMA* class), in an analogy to the Seasonal Autoregressive Integrated Moving Average models (SARIMA) extension of ARMA models, thus including a multiplicative seasonal autoregressive integrated moving average model. Their estimate is carried out through a full Bayesian inference procedure, on the grounds of a weakly stationary model assumption and consequently constraining of the autoregressive and moving average parameters. The outcome assumes a Negative Binomial distribution, whereas the parameters follow Beta, Gaussian and Gamma priors. The model is subsequently applied to a Malaria time series analyses.

Andrade, Leslow and Andrade (2016) proposed a Transformed *GARMA* model to cope with non-additivity, non-normality and heteroscedasticity in time series; the transformation ensures that the transformed series fulfill the *GARMA* model assumptions. Andrade, Andrade and Ehlers (2016) also estimate this model under the Bayesian framework and a simulation study is carried out followed by an analysis of fertility rates in Sweden.

Additionally, Andrade, Ehlers and Andrade (2016) developed a Bayesian *GARMA* model for count data and applied it to three Brazilian datasets, one for automobile production, other for dengue fever hospitalizations and another for number of deaths by dengue. They used a Poisson, Binomial and Negative Binomial *GARMA* models, with multivariate Gaussian priors, though non-informative (large variances resulting in flat densities), for each model parameter.

Zheng, Hiao and Chen (2015) propose an extension of the *GARMA* model, the *martingalized GARMA* model (M-*GARMA*), in which the resulting transformed ARMA model (through an appropriate *link function*) has a martingale difference sequence as its error sequence. This property is only achieved in the original model (Benjamin et al., 2003) in the case of the identity link function. The improvement of this new model is striking, as maximum likelihood asymptotic distribution can be established. Simulations for a *Log-Gamma-M-GARMA* and *Logit-Beta-M-GARMA* models are performed, followed by an application to High-frequency realized volatility making usage of the *Gamma-M-GARMA* model with logarithmic *link function* and another application studying the US personal saving rate through the *Logit-Beta-M-GARMA.*

It is worth noting that the *GARMA* applications in the literature generally focused on the discrete type of distributions, therefore we try to fill this gap applying the model to a continuous financial time series. From the empirical finance point of view, time series such as assets returns

(difference of log prices) usually exhibit some stylized facts that result in a departure from the Gaussian distribution assumption. Some of those facts are returns low serial correlation and volatility (i.e. squared returns, as usually returns are centered at zero) clustering and asymmetry.

In this line of thought, Engle (1982) in his pioneer work, developed the autoregressive conditional heteroscedasticity model, where returns are normally distributed but the volatility process is from an autoregressive nature. Thus, he created a simple model that can cope with variance changing over time as a function of past errors, whilst the unconditional variance remains constant through time. Remarkably, in 1986, Tim Bollerslev extends Engle's model and introduces the class of Generalized Autoregressive Conditional Heteroscedasticity (GARCH) models (Bollerslev, 1986), in the same sense that an ARMA process extends an AR one. Small adaptations or simplifications to this model also took place; take the ARCH-M (Engle et al., 1987) or TARCH (Glosten et al., 1993) for instance.

In this work we will take advantage of the relation between the Gamma-*GARMA* and GARCH models, as presented in the paper of Benjamin et al. (2003), to analyze a financial time series. Our real data examples bear a close resemblance to one of the applications present in the work of Zheng, Hiao and Chen (2015), the difference being that we will not attempt to model intraday high-frequency volatility, only daily prices keeping in mind the GARCH equivalence.

2.2 BOOTSTRAP methods

The term bootstrap was coined by Bradley Efron in his seminal paper "Bootstrap Methods: Another Look at the Jackknife", Efron (1979), where the author introduces the bootstrap methodology and shows that the nonparametric jackknife can be viewed as a linear approximation method for the bootstrap. In this work, Efron focused on estimating the sampling distribution of a given statisic by its bootstrap distribution. This new tool is more widely applicable than the jackknife, useful to estimate parameters bias, variance and also to construct confidence intervals.

The problem of calculating the bootstrap distribution can be tackled by three methods: direct theoretical calculation, Monte Carlo approximation and Taylor series expansion. The second approach is often employed as it is easier to implement. The author also discusses the problem of error rate estimation in the case of discriminant analysis and shows that the bootstrap method outperforms the leave-one-out cross-validation (a method for estimating error rates by

leaving one observation out of the estimation process at a time and afterwards using it as independent for estimating the error measure) approach.

In theory, the bootstrap scheme is really simple, that is, in possession of the bootstrap resamples what one really has to do is to approximate the sampling distribution of the statistic of interest by its bootstrap one. Therefore, the tough part of this algorithm is how to compute the so-called bootstrap distribution. Due to its popularity and simplicity, the Monte Carlo approximation method is the one followed here.

Efron (1980) gives a thorough account of the relation between the Jafkknife, Bootstrap, cross-validation, balanced repeated replications and random subsampling. Additionally, some nonparametric confidence intervals are employed, namely the percentile method, the bias-corrected percentile method and the t-bootstrap.

The matter of error rate estimation is developed further in Efron (1983), where the relation between the bootstrap and cross-validation estimation is analyzed. Some interesting ideas are proposed such as the double bootstrap and the 0.632 estimator. Subsequently, estimates of the downward bias of the apparent error rate are provided in Efron (1986), in which a theory in the GLM framework is stated. He also compares Mallow's C_p, cross-validation, generalized cross-validation, bootstrap and Akaike's information criterion (AIC).

Moving beyond Efron's work, we cannot understate the monograph of Hall (1992). It not only provides a systematic review of bootstrap methods but also a rigorous mathematical evaluation of the bootstrap performance through the usage of Edgeworth expansions.

Despite its simple nature, when it comes to the Monte Carlo approximation to the bootstrap distribution in the time series framework some difficulties or specificities arise. This happens because the original bootstrap was conceived to deal with independent datasets and from the time series point of view this is an overly simplistic assumption (as correlation is frequently induced).

From the time-series point of view, our work relies heavily on the work of Chernick (2008) and Chernick and LaBudde (2011). Both books give a description of the bootstrap and its relation to parameter bias, location and dispersion estimation. They also handle with confidence intervals and hypothesis testing. Also, and perhaps more important to our analysis, a survey of bootstrap methods in the time series framework is given. The content of the books concerning this topic is quite similar, with both covering the basics of model-based and block resampling bootstrap, the main difference between them being some additional bootstrap schemes in the newer book, namely the Dependent Wild Bootstrap.

Additionally, the book Resampling Methods for Dependent Data, from Lahiri (2003), is a keen source of information, where the author reviews some Block Bootstrap Methods, establishing their consistency, second-order properties, contrast their performance, he also approaches the problem of resampling methods for spatial data. Nonetheless, for this work, we highlight the importance of the chapter on the Empirical Choice of Block Size.

The resampling methods in the time-series context usually fit two main categories: model-based or the block resampling. For completeness this chapter has two subsections dedicated to the aforementioned resampling schemes, followed by a third subsection with some methods that do not fall in these categories.

2.2.1 Model-Based Bootstrap

Model-based bootstrapped time-series consists of assuming a model, isolating its residuals in the model equation and bootstrapping the residuals. Thus, the bulk of model-based methods is the resample of the model residuals and, because of this, an explicit form of residual in the model equation is required (Chernick and LaBudde 2011 p. 118). Consequently, this is a model dependent method, in which its validity depends on the correctness of the specified model. The prime example in the literature is the first order autoregression, where an estimate of the autoregressive coefficient (usually through a Maximum Likelihood Estimation) is used in computing the model residuals. However, this is an overly simplified structure and more complex models are hard to handle in a similar fashion. See for example Efron and Tibshirani (1986), Shao and Tu (1995), Chernick (2008) or Chernick and LaBudde (2011).

A review of bootstrap ideas and applications can be found in Efron and Tibshirani (1986). From all examples present in their text, the more relevant to our work is the first-order autoregressive one, introducing the notion of bootstrap residuals in the times series context. The authors also introduce several approaches to calculate confidence intervals, such as the standard, percentile, bias-corrected percentile and BC_{α} methods.

Any of the methods described in the previously stated references in the case of model-based bootstrap can be adapted into the *GARMA* context, the difference being that this is a dual stage process. In the first step the residuals are computed (e.g.: original scale, Pearson, predictor scale) and sampled with replacement entering the linear predictor. In step two, a random sample is drawn from the respective *GARMA* distribution with mean given by the inversion of the *link*

function evaluated at step 1. This process is repeated sequentially until the original series length is recovered.

However, it is worth noting that this kind of bootstrap procedure is hard to develop in a general GARMA setting, as it will be seen in Section 4. Unlike ARMA models that can have an infinite autoregressive or moving average representation and consequently are prone to bootstrap residuals, GARMA models do not have this type of representation and cannot be directly bootstrapped in a general model-bootstrap perspective. Due to these aforementioned restrictions, in this work we will restrain ourselves to another class of bootstrap resampling in time series, known as Moving Block Bootstrap (MBB).

2.2.2 Block-resampling Bootstrap

Block-resampling bootstrap has been designed to deal with the model misspecification pitfall. Chernick and LaBudde (2011) pointed out that the moving block bootstrap has been the most successful attempt in the time domain approach. It was introduced by Carlstein (1986) and further developed by Kunsch (1989). Some block resampling methods described in the book and documented by Lahiri (2003) are:

> The various types of block bootstrap approaches covered by Lahiri include(1) MBB, (2) nonoverlapping block bootstrap (NBB), (3) circular block bootstrap (CBB), (4) stationary block bootstrap (SBB), and (5) tapered block bootstrap (TBB)(very briefly) (Chernick and LaBudde, 2011 p.124).

A theoretical comparison of the MBB, NBB, CBB and SBB can be found in Lahiri (1999). The simulation results show that for a moderate sample size the MBB and CBB are preferable over the NBB and SBB.

Lahiri (2003, p. 206) simulated two confidence intervals for the autoregressive bootstrap (ARB), that is a p-order autoregression, and the moving block bootstrap (MBB) with four different block lengths. Theoretically, this is a situation where ARB should perform better than the MBB; however, for some block lengths the MBB gets close to the ARB. Thus, even though the MBB is expected to have a poor performance when contrasted to the ARB it is in fact more robust to model misspecification and might be better suited in cases where there is uncertainty about the model correctness.

It is evident that there are many block bootstrapping schemes, though we focus mainly on the MBB owing to its history of success and consistency property in the specific case of

GARMA models. Remarkably, despite the low number of papers based on the *GARMA*, Andrade (2016) established the assumptions guaranteeing the consistency of the MBB for this specific model. Here, consistency means that for an increasing sample size, the quantiles produced by the MBB converge to the quantiles of the respective asymptotic distribution. One disadvantage of the MBB is the heuristic nature of the block length selection process.

To tackle the block length issue, some estimators have been proposed, such as the one by Hall, Horowitz and Jing (1995) (HHJ) and by Lahiri, Furukawa and Lee (2007) (generalized plug-in rule or nonparametric plug-in method). The latter is based on a Jackknife-After-Bootstrap (JAB) method while in the former the length depends on the context and can be a simple function of the sample size. For a comparison of them, please refer to the original paper of Lahiri, Furukawa and Lee (2007).

2.2.3 Other resampling procedures

Aside from model-based and block resampling schemes other procedures have been proposed, though, a thorough literature reviewing those topics is beyond the scope of this monograph and the reader might refer to Chernick (2008) or Chernick and LaBudde (2011), for instance. Only a brief description of two methods is provided below.

One alternative to the model based and block resampling bootstrap is the Dependent Wild Bootstrap (DWB), proposed by Shao (2010), that extends the wild bootstrap to the case of stationary, weakly dependent, time series. No partitioning of the data into blocks is required and it is applicable in the case of irregular time series. The method relies on the DWB pseudo observations, which are simply a function of sample statistics.

Moving onto the *GARMA* – GARCH relation we might have to keep in mind the intrinsic properties of assets returns, and as such, some bootstrap resampling methods might be unsuitable. Vinod (2004) had put forth three properties that might render the traditional bootstrap inappropriate and developed the Maximum entropy bootstrap (MEB) that can cope with those drawbacks simultaneously. The latter is also simplified and extended into a panel-data setting in Vinod (2006). The MEB comes as an alternative to the block resampling methods and does not demand the block subsetting of the data. It is more general than the MBB, since it does not require the stationarity assumption and does not need differencing (in an ARMA context).

3 THE *GARMA* MODEL

Let *Y* be a stochastic process, i.e. a collection of random variables $Y = \{Y_t(\omega), t \in \mathcal{T}, \omega \in \Omega\}$, defined in the probability space $(\Omega, \mathcal{F}, P)$, where Ω is the set of all possible states, $\mathcal{F}$ is a σ-*field* of all subsets of Ω, *P* is a probability measure under $\mathcal{F}$ and $\mathcal{T}$ an arbitrary set. We have for a fixed $t \in \mathcal{T}$ and for each fixed value of $\omega \in \Omega$ that $y_t(\omega)$ is a realization or path of the process. Also, $Y_t(\omega)$ is a random variable for each t and a fixed ω and for simplicity the index ω will be subsumed.

The former definition of *Y* is too general for the *GARMA* model (Benjamin et al., 2003) and some simplifications can be adopted. It is a discrete time process so $\mathcal{T}$ is a finite or enumerable set and is taken as the set of integers $\mathbb{Z}$ $(t \in \mathbb{Z})$. Additionally, the process can be redefined in the filtered probability space, $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, P)$, where $\{\mathcal{F}_t\}_{t\geq 0}$ is a filtration. Here, filtration is defined as an increasing sequence of sub σ-*fields* on the measurable space $(\Omega, \mathcal{F})$, that is, $\mathcal{F}_t \in \mathcal{F}$ and for $t_1 > t_2 \Longrightarrow \mathcal{F}_{t_1} \subseteq \mathcal{F}_{t_2}$. Leaving mathematical technicalities aside, one can think of $\mathcal{F}_t$ as the information set available at time t including all the previous information until t.

In terms of the *GARMA* model we have that each realization of $Y_t, t = 1, \dots, n$, has a conditional distribution belonging to the same exponential family. The conditioning is with respect to $\mathcal{F}_{t-1}$, and in this case $\mathcal{F}_{t-1} = \{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}; y_1, \dots, y_{t-1}; \mu_1, \dots, \mu_{t-1}\}$. Thus, the conditional density of $Y_t|\mathcal{F}_{t-1}$ is of the form:

$$f_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) = exp\left\{\frac{y_t\vartheta_t - a(\vartheta_t)}{\varphi} + b(y_t, \varphi)\right\} \qquad (1)$$

where $a(\cdot)$ and b(·) are specific functions defining the particular member of exponential family, with ϑ_t as the canonical and φ as the scale parameters, **x** is a *r* dimensional vector of explanatory variables and μ is the mean vector. From standard GLM results (McCullagh and Nelder, 1989) it can be shown that the term $\mu_t = a'(\vartheta_t) = E_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1})$ and $Var_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) = \varphi a''(\vartheta_t)$, here $'$ and $''$ denote the first and second derivatives of $a(\cdot)$, respectively.

Moreover, the predictor η_t is such that $\eta_t = g(\mu_t)$ and g is the link function (a one-to-one monotonic function), in resemblance to the GLM terminology. Parameter η_t can be generally defined but the following flexible and parsimonious submodel is more appropriate (than the generally defined one):

$$\eta_t = \mathbf{x}_t'\boldsymbol{\beta} + \sum_{j=1}^{p} \phi_j\{g(y_{t-j}) - \mathbf{x}_{t-j}'\,\boldsymbol{\beta}\} + \sum_{j=1}^{q} \theta_j\{g(y_{t-j}) - \eta_{t-j}\} \qquad (2)$$

where $\boldsymbol{\beta}' = (\beta_1, \beta_2, \dots, \beta_r)$, is the vector of parameters of the linear predictor $\mathbf{x}_t'\boldsymbol{\beta}$ of η_t, $\boldsymbol{\phi}' = (\phi_1, \phi_2, \dots, \phi_p)$ the vector of autoregressive parameters, $\boldsymbol{\theta}' = (\theta_1, \theta_2, \dots, \theta_q)$ the vector of moving average parameters. Equations (1) and (2) together define the *GARMA* model.

In the following two subsections, the Poisson and Gamma *GARMA* models are defined. Note, however, that the *GARMA* class is not limited to these models and can be applied to any member of the exponential family.

3.1.1 The *POISSON – GARMA* model

If $Y_t|\mathcal{F}_{t-1}$follows a Poisson distribution with mean parameter μ_t then its p.m.f. is:

$$f_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) = \frac{e^{-\mu_t}{\mu_t}^{y_t}}{y_t!} = exp\{y_t \log(\mu_t) - \mu_t - \log(y_t!)\}, \quad y_t = 0,1,2, \dots. \qquad (3)$$

It is evident that $Y_t|\mathcal{F}_{t-1}$belongs to the exponential family of distributions and also that $\vartheta_t = \log(\mu_t)$, $a(\vartheta_t) = e^{\mu_t}$, $b\,(y_t, \varphi) = -\log(y_t!)$ and $\varphi = 1$. The canonical *link function* in this case is $\log \equiv ln$. Hence, η_t is such that:

$$\eta_t = \log(\mu_t) = \mathbf{x}_t'\boldsymbol{\beta} + \sum_{j=1}^{p} \phi_j\{\log(y_{t-j}^*) - \mathbf{x}_{t-j}'\,\boldsymbol{\beta}\} + \sum_{j=1}^{q} \theta_j\{\log(y_{t-j}^*/\mu_{t-j})\}, \qquad (4)$$

where $y_{t-j}^* = \max(y_{t-j}, \alpha)$, $0 < \alpha < 1$. Here α is taken as 0.1. As previously, the *Poisson-GARMA* model is defined by equations (3) and (4).

3.1.2 The *GAMMA – GARMA* model

Likewise, if $Y_t|\mathcal{F}_{t-1}$ follows a Gamma distribution with shape parameter δ and scale parameter γ (so a $\Gamma(\delta,\gamma)$ distribution), thus, its p.d.f. is given by:

$$f_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) = \frac{{y_t}^{\delta-1}e^{-y_t/\gamma}}{\Gamma(\delta)\gamma^{\delta}}. \qquad (5)$$

with $E_{Y_t|\mathcal{F}_{t-1}} = \delta\gamma$ and $Var_{Y_t|\mathcal{F}_{t-1}} = \delta\gamma^2$. However, there is a more useful re-parametrization of the Gamma density that makes it better suited to be applied in the *GARMA* model. Let $\delta = 1/\sigma^2$ and $\gamma = \sigma^2\mu_t$, so that $E_{Y_t|\mathcal{F}_{t-1}} = \mu_t$, $Var_{Y_t|\mathcal{F}_{t-1}} = \sigma^2\mu_t^2$ and the transformed p.d.f. is equivalent to:

$$f_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) = exp\left\{\frac{1}{\sigma^2}\left(-\frac{y_t}{\mu_t} - log(\mu_t)\right) + \left[-\frac{log(\sigma^2)}{\sigma^2} - log\left(\Gamma\left(\frac{1}{\sigma^2}\right)\right) + log(y_t)\left(\frac{1}{\sigma^2}-1\right)\right]\right\}. \qquad (6)$$

Thus, $Y_t|\mathcal{F}_{t-1}$ belongs to the exponential family of distributions with $\vartheta_t = -\frac{1}{\mu_t}$, $a(\vartheta_t) = log(\mu_t)$, $b(y_t,\varphi) = [\cdot]$ and $\varphi = \sigma^2$. The canonical *link function* in this case is the reciprocal function, though, for simplicity, $g(\mu_t)$ is taken as $\log(\mu_t)$. Hence, η_t is the same for the Gamma and Poisson model, so that equations (6) and (4) define the *Gamma-GARMA* model, while here $y^*_{t-j} = y_{t-j}$.

3.1.3 Maximum likelihood estimation of model parameters

In the *GARMA* model, after observing a sample with $\mathbf{x}_1,\dots,\mathbf{x}_t; y_1,\dots,y_t$, one can estimate the given model parameters $\boldsymbol{\beta}'$, $\boldsymbol{\phi}'$ and $\boldsymbol{\theta}'$ through the method of maximum likelihood. The likelihood of the model $L(\boldsymbol{\beta},\boldsymbol{\phi},\boldsymbol{\theta})$ and the log-likelihood $l(\boldsymbol{\beta},\boldsymbol{\phi},\boldsymbol{\theta}) = \log(L(\boldsymbol{\beta},\boldsymbol{\phi},\boldsymbol{\theta}))$ can be defined as:

$$L(\boldsymbol{\beta},\boldsymbol{\phi},\boldsymbol{\theta}) = \prod_{t=1}^{n} f_{Y_t|\mathcal{F}_{t-1}}(y_t|\mathcal{F}_{t-1}) \qquad (7)$$

$$l(\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{t=1}^{n} \log f_{Y_t|\mathcal{F}_{t-1}}\left(y_t|\mathcal{F}_{t-1}\right), where\ log = ln. \qquad (8)$$

As log is a one-to-one monotonic function the value that maximizes $L(\cdot)$ is the same that maximizes $l(\cdot)$, and for computation simplicity the log-likelihood (LLH) estimates of the parameters are computed. The maximum likelihood estimates (MLE) are such that: $LLH = l\left(\widehat{\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta}}\right)$, where $\widehat{\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta}} = \underset{\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta}}{\operatorname{argmax}}\, l(\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta})$. This task is carried out through a numerical optimization routine.

3.1.4 Some simulated models

For illustration purpose, in this section, some models were simulated and estimated through maximum likelihood. First, consider 1000 simulations of a series of length 1000 of a *Poisson – GARMA* model, with an autoregressive term of value 0.15 ($\phi_1 = 0.15$) and constant intercept equals to 2. Figure 1 depicts the Monte Carlo MLE empirical distribution of the model parameters. The mean of ϕ_1 estimates is 0.14, while the true value is 0.15. If we compute an empirical 95% symmetrical confidence interval we have that $\hat{\phi}_1 \in [0.08, 0.19]$, a good result, as we should expect the true value of the parameter to be included in this interval. For β, the simulations show a fairly accurate estimate, where the mean of the simulations is 1.9991, contrasted to the true value of 2, which belongs to the 95% empirical confidence interval $(1.97, 2.02)$.

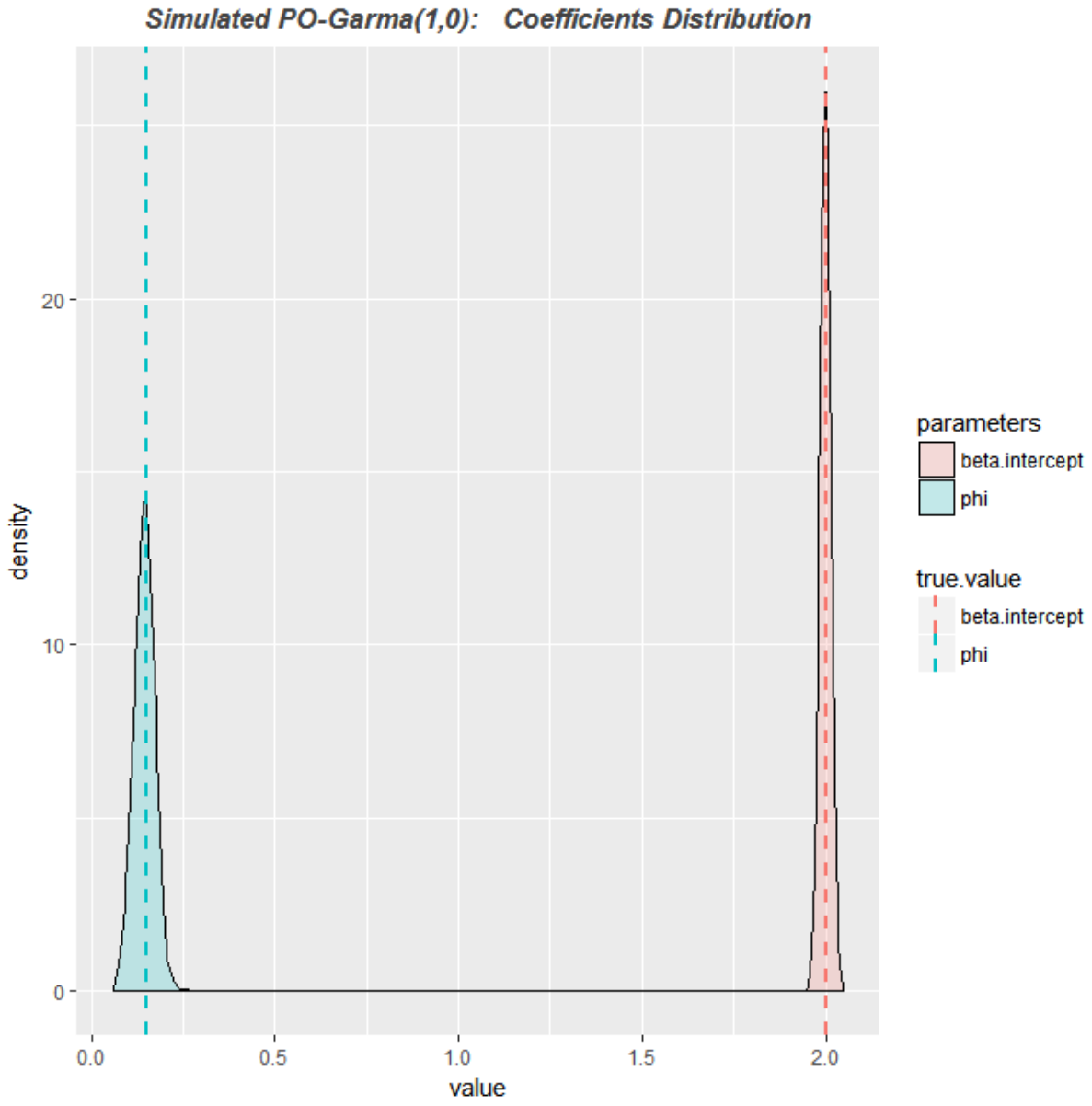


Figure 1: *Poisson - GARMA* (1, 0), $\phi_1 = 0.15, \mathbf{x}_t'\boldsymbol{\beta} = \mathbf{x}_{t-1}'\boldsymbol{\beta} \equiv 2$, 1000 simulations MLE parameters empirical distribution

Second, we shall examine a model with more parameters and check whether this additional complexity can be well captured by the estimation process. As in the last example take into consideration 1000 simulations of a series of length 1000 of a *Poisson – GARMA* model, with an autoregressive term of value 0.50 ($\phi_1 = 0.50$), moving average term of 0.1 ($\theta_1 = 0.1$) and constant intercept equals to 2. Figure 2 shows the parameters MLE empirical distribution. The mean estimate of ϕ_1 is 0.48 with an empirical 95% symmetrical confidence interval $(0.39, 0.56)$. Likewise, in the case of θ_1, the mean of the Monte Carlo MLE estimates is 0.07 with the respective 95% confidence interval $(0.02, 0.16)$. The same reasoning follows

for β, with mean value of 2.0001 and interval with inferior and superior limits of 1.94 and 2.05, respectively.

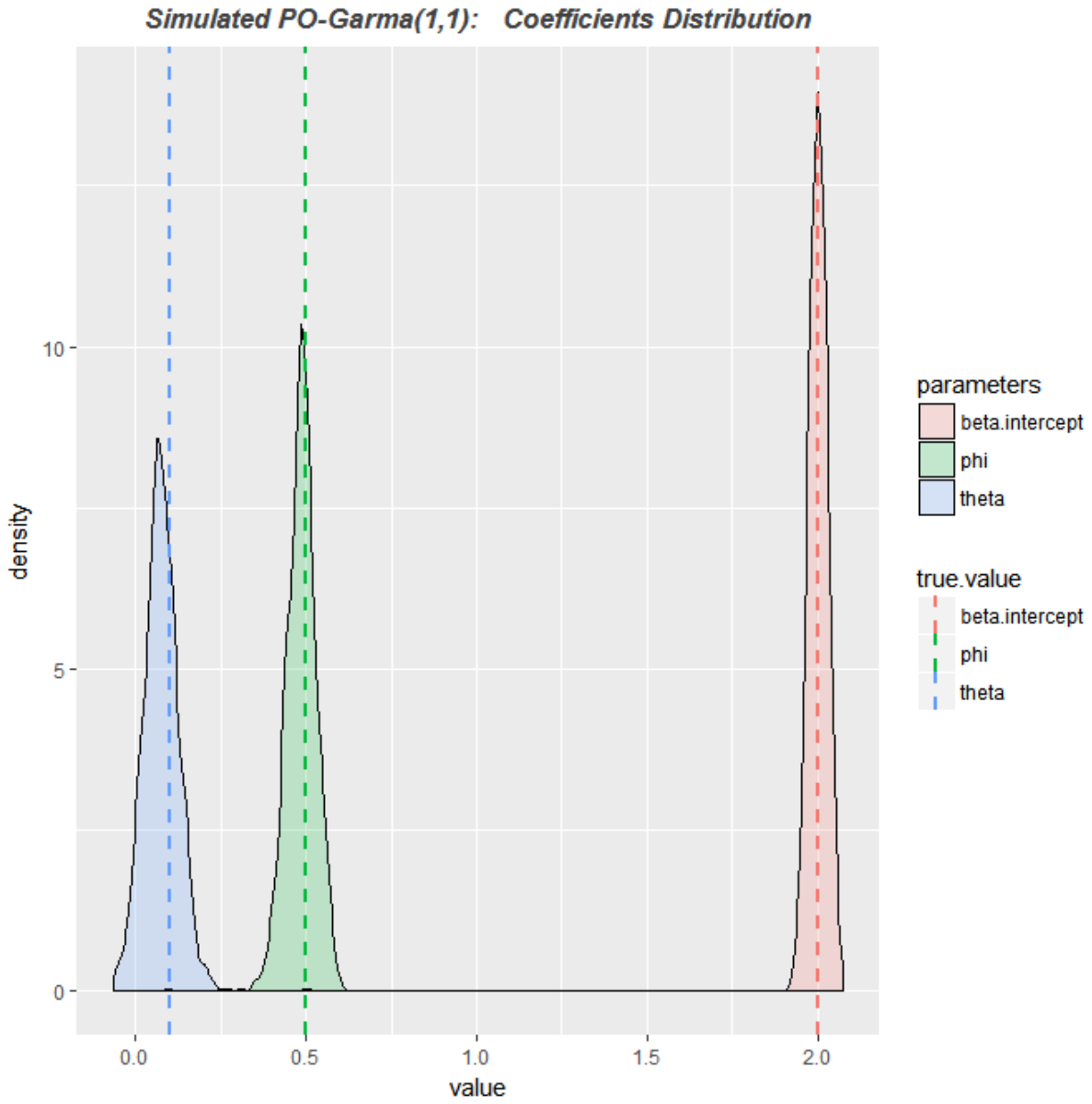


Figure 2: *Poisson - GARMA* (1, 1), $\phi_1 = 0.50, \theta_1 = 0.1, \mathbf{x}_t'\boldsymbol{\beta} = \mathbf{x}_{t-1}'\boldsymbol{\beta} \equiv 2$, 1000 simulations MLE parameters empirical distribution

Third, we consider a similar order *Gamma – GARMA* model with simulation characteristics similar to the last example, the difference being the additional parameter $\sigma^2 = 2$. This model is exhibited in Figure 3. Again, we see that the parameter estimates behave well, in the sense that all true values belong to their empirical 95% confidence interval. When contrasted to the previous models, the absolute difference between the mean values and true values are higher. This is a family specific phenomena, that is, the *Poisson* family *GARMA*

models are easier to estimate while the *likelihood* function of the *Gamma* is more complex, making the numerical optimization problem harder and thus more prone to failure and/or numerical instability. Consequently, this family behavior seems to worsen the overall estimation results of the *Gamma – GARMA* models when contrasted to their *Poisson* counterpart.

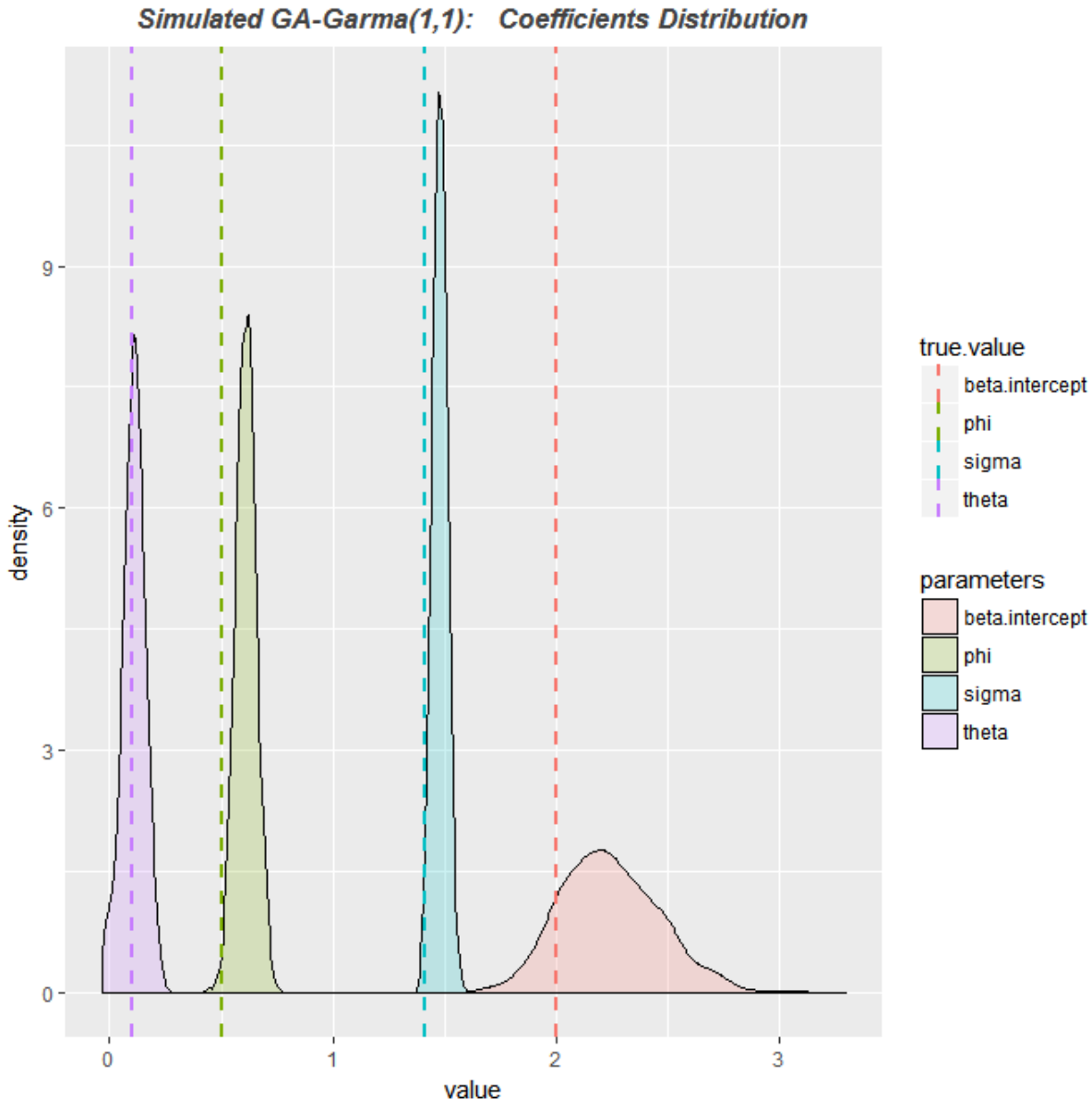


Figure 3: *Gamma - GARMA* (1, 1), $\phi_1 = 0.50, \theta_1 = 0.1$, $\sigma^2 = 2$, $\mathbf{x}_t'\boldsymbol{\beta} = \mathbf{x}_{t-1}'\boldsymbol{\beta} \equiv 2$, 1000 simulations MLE parameters empirical distribution

Lastly, in Figure 4, we examine the effect of increasing the coefficient absolute value in the first example, adopting $\phi_1 = 0.80$. In this case, the mean of ϕ_1 estimates is 0.72, while the true value is 0.8. The empirical 95% symmetrical confidence interval is$(0.67, 0.76)$, a poor

result, as we should expect that the true value of the parameter to be included in this interval. Additionally, the maximum of the distribution is 0.79, a value smaller than the true parameter value. This relative poor performance might be explained due to the high value of the simulated model parameter, leading to non-stationarity of the simulated *GARMA* process. For β, the simulations show a fairly accurate estimate, where the mean of the simulations is 2.02, contrasted to the true value of 2, with the true value belonging to the 95% empirical confidence interval.

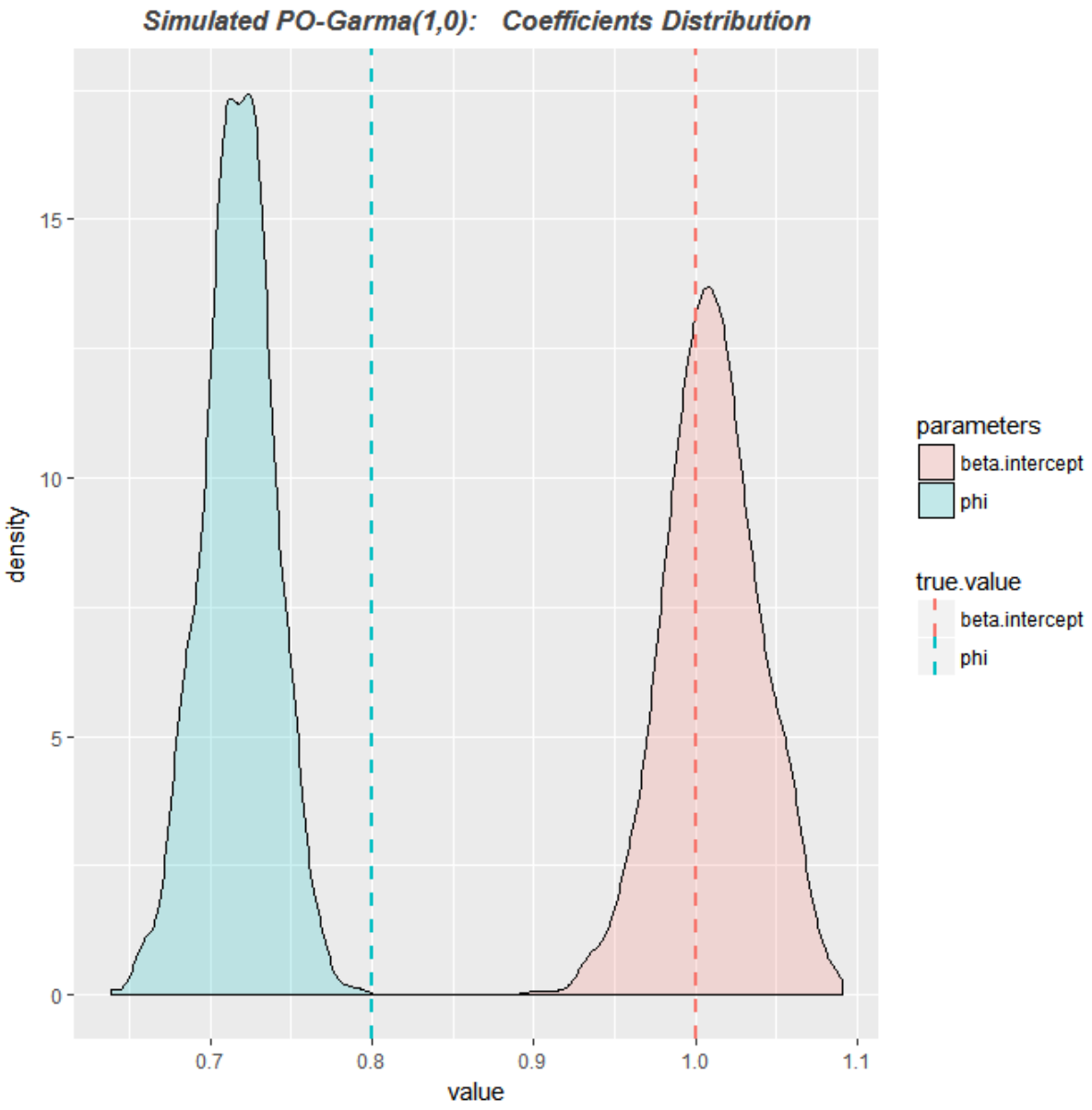


Figure 4: *Poisson - GARMA*(1,0), $\phi_1 = 0.80, \mathbf{x}_t'\boldsymbol{\beta} = \mathbf{x}_{t-1}'\boldsymbol{\beta} \equiv 2$, 1000 simulations MLE parameters empirical distribution

The last example raises an important issue regarding the adequacy of the maximum likelihood estimates in the case of *GARMA* models when $|\boldsymbol{\phi}| \cong 1$ or $|\boldsymbol{\theta}| \cong 1$. For this reason, all subsequent estimated models avoid this problem by restricting the simulated parameter to values smaller than or equal to 0.5 in absolute value.

3.1.5 Additional Properties

In this section, some properties regarding the stationary conditions for the marginal mean and variance and the stationary mean and variance of Y_t are supplied for the case where the link function g is the identity function. These properties are provided in the work of Benjamin et al. (2003). The marginal mean of Y_t is given by:

$$E_{Y_t}(y_t) = \mathbf{x}_t'\boldsymbol{\beta} \tag{9}$$

conditional on the invertibility of $\Phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$. Thus, the marginal mean is stationary provided also that $\mathbf{x}_t'\boldsymbol{\beta} = \beta_0$ for all t. The marginal variance is:

$$Var_{Y_t}(y_t) = \varphi E_{Y_t}[\psi^2(B)v(\mu_t)] \tag{10}$$

where $\psi^2(B) = 1 + \psi_1^2 B + \psi_2^2 B^2 + \cdots$ and $\psi(B) = \Phi(B)^{-1}\Theta(B) = 1 + \psi_1 B + \psi_1 B^2 + \cdots$, under the assumption that $\Phi(B)$ is invertible and $\Theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$.

In the specific case of the *Poisson – GARMA* model we have that:

$$Var_{Y_t}(y_t) = \psi^2(1)\beta_0 \tag{11}$$

where $\psi^2(1) = 1 + \sum_{j=1}^{\infty} \psi_j^2$.

Whilst, for the *Gamma – GARMA* model:

$$Var_{Y_t}(y_t) = \varphi\, \psi^2(1)\left[1 + \varphi + \varphi\psi^2(1)\right]^{-1} \beta_0^2 \tag{12}$$

provided that $[1 + \varphi + \varphi\psi^2(1)]$ is invertible. All the proofs for the previously stated results can be found at Benjamin et al. (2003).

Despite being readily available, stationarity results for identity link function are not general enough. As pointed out by Woodard et al. (2011), the latter case excludes many popular count-valued models, thus, a more general approach must be followed. Additionally, Woodard et al. (2011) provide strict stationarity conditions for *GARMA* models in the absence of covariates (the term $\mathbf{x}_t'\boldsymbol{\beta}$).

4 MOVING BLOCK BOOTSTRAP IN GARMA MODELS

In the specific case of the *GARMA* model, Equation (2) shed some light into what is needed to construct a Model-Based bootstrap scheme. The only possible way to do it is when the moving average term is present, because only on it there is a residual term being formed. Thus, this restriction significantly reduces the range of applicability of the model-based bootstrap to the *GARMA* model, as pure autoregressive models are no longer feasible (if the purpose is to use this bootstrap method).

In contrast to the Model–Based scheme, the Moving Block Bootstrap (MBB) method for stationary processes builds on the idea that while successive observations are correlated, observations separated far enough in time will be approximately uncorrelated and can be treated as exchangeable (Chernick, 2008 p. 104).

For instance, consider the sample $\mathbf{y} = y_1, \dots, y_n$, a series of length n and suppose that $n = bL$, where b denotes the number of overlapping blocks and L the respective block length and both are positive integers, that is, $b \in \mathbb{Z}^+$ and $L \in \mathbb{Z}^+$. The MBB (Kunsch, 1989) consists of sampling b blocks with replacement from the $n - L + 1$ blocks to generate the sequence $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_n^*)$ of bootstrap resample, this process being repeated B times.

Some care must be taken with the MBB as observed by Chernick:

> Some of the drawbacks of block methods in general are as follows: (1) Resampled blocks do not quite mimic the behavior of the time series, and (2) they have a tendency to weaken the dependency in the series. (Chernick, 2008 p. 105)

Those downsides are directly related to the selection of the optimal block length. A higher value of L is associated with a reduction in the bias of the bootstrap parameter estimation, when contrasted to a smaller value, as the replicates will more closely resemble the original series. So, the stronger the dependencies in Y, the higher L should be. Conversely, a

smaller value of L translates into a variance reduction in the estimation as more replicates are available. (Carlstein, 1986 p. 1176)

This Section is divided in two Sub-Sections, the first approaches the optimal block length issue and the second the bootstrap bias correction and confidence interval estimation process.

4.1 On the optimal block length choice

In this section, some remarks concerning the optimal block length choice in the context of MBB and *GARMA* models are made. Here, optimality is assessed in terms of closeness of the MBB resamples mean to the parameters true values. That is, the optimal length $\left(L^{Optim}\right)$ for a given parameter ζ is defined as:

$$L^{Optim}(\zeta) = \min_{L} abs\left(\zeta - \frac{1}{B}\sum_{i=1}^{B} \hat{\zeta}_{MLE}^{*;L;\,(i)}\right), L \in \mathbb{L}, \qquad (13)$$

where $\mathbb{L}$ is the set of block lengths on which optimality is being evaluated. Here, abs is the absolute value function and ζ represents any estimated parameter of a *Poisson* or *Gamma GARMA* model. It is worth noting that the optimal block length for a model is parameter specific, that is, the optimal value might be different for each parameter in the model. One could also derive a global measure of optimality ($L_{G.}^{Optim}$) by simply computing the mean absolute deviation for all parameters and all $L \in \mathbb{L}$ and choose the one with the smallest average, that is

$$L_{G.}^{Optim}(\boldsymbol{\zeta}) = \min_{L} abs \sum_{k \in \mathcal{K}} \left(\zeta_k - \frac{1}{B}\sum_{i=1}^{B} \hat{\zeta}_{k,MLE}^{*;L;\,(i)}\right), L \in \mathbb{L}, \qquad (14)$$

where $\boldsymbol{\zeta}$ is the vector of model parameters and $\mathcal{K}$ is the set of all parameter values in the specified model. $L_{G.}^{Optim}$ is a reasonable metric if each L^{Optim} in $\boldsymbol{\zeta}$ is close to one another, in other words, $L_{G.}^{Optim}$ is good if the variance of the optimal block lengths for each parameter in the model is small and around the same value.

In the case of a small series, with length 30 for instance, $\mathbb{L}$ can be taken as the set of the integers from one to thirty, where optimality for a specific model parameter can be precisely computed (if we are working with a simulated series or know *ex-ante* the true data generating

process and thus the parameters true values). On the other hand, in a large series, say with $n = 1000$, computing all possible block lengths might be a cumbersome task. It is feasible, though it might take too much time. In this case, a smaller number of possible block lengths might be evaluated and optimality assessed heuristically. If $\hat{\zeta}_{k,MLE}^{*;L;\,(i)}$ is a monotone function of L, an equally spaced grid of possible values of $L \in \mathbb{L}$ might give a first global measure of optimality and in a second step the grid might be evaluated in the region of better predictions in the first step, thus narrowing the search for L^{Optim}.

Some of the approaches proposed in the literature employ a squared loss function, and the optimal search is measured with respect to $\hat{\zeta}_{MLE}$. In those instances, the heuristic provided in the previous paragraph might be helpful.

As previously stated, optimality can be locally or globally defined, which might lead to non-uniqueness in the chosen optimal block length. Additionally, the optimal block length might not be the same for bias and distribution function estimation. This, in turn, makes our task more difficult, as parameter bias and confidence interval estimation might require different optimal block lengths. For conciseness, optimality is only evaluated with regard to bias estimation, but the reasoning is easily extended to the confidence interval case.

Hall, Horowitz and Jing (1995) proposed some rules for identifying the optimal block length in the bootstrap with dependent data. They point out that the optimal block length depends on the context, being of order equal to $n^{1/3}$ for bias and variance estimation, $n^{1/4}$ in the case of one-sided distribution function and $n^{1/5}$ for the two-sided case. These results following a squared loss function. They also propose an empirical method for choosing the block length. This procedure is reviewed in Lahiri (2003) and explained below.

Let $\hat{L}_n$ denote the estimate for the optimal block size for the entire series (given the statistic $T(\mathbf{y})$ of interest) and $\hat{L}_m, m < n$, the optimal value for a series of smaller length than the original one. Then, $\hat{L}_n = (n/m)^{1/k}\ \hat{L}_m,\ for\ k = 3, 4, 5$, where k is determined by the context (bias/ variance, one-sided or two-sided distribution functions). Denote by $\mathbb{S}$ the set of all subseries of length m from $\mathbf{y}$. Apply the MBB to each element of $\mathbb{S}$, with $L' \in \mathbb{Z}_+^m$, where L' is the block length value and $\mathbb{Z}_+^m$ is the set of all positive integers until m (we might take all values in $\mathbb{Z}_+^m$ or a smaller subset, the choice might rely on the computational burden), leading to $\hat{T}_{L'}^{\mathbb{S}}$ ($\hat{T}_{L'}^{\mathbb{S}}$ is the value of the statistic of interest computed with all the elements of $\mathbb{S}(m)$). Compute $\hat{T}_{L'}^{n}$ (the statistic evaluated at the entire data set) and then the estimate of the mean

squared error$\left(\widehat{MSE}\right)$ given by: $\widehat{MSE} = \sum_{\mathbb{S}' \in \mathbb{S}} \left(\hat{T}_{L'}^{n} - \hat{T}_{L'}^{\mathbb{S}'}\right)^2 \frac{1}{n-m+1}$. Take $\hat{L}_m = \underset{L'}{\operatorname{argmin}}\left(\widehat{MSE}\right)$, with $L' \in \mathbb{Z}_{+}^{m}$ and obtain $\hat{L}_n$. This process can be (and in this monograph is) iterated.

Building on the work of Hall, Horowitz and Jing (1995), Lahiri, Furukawa and Lee (2007) developed a nonparametric plug-in rule (**NPPI**), based on the Jackknife-After-Bootstrap (Lahiri, 2002), which is consistent not only for bias, variance and distribution estimation but also for bootstrap quantile estimation. In their approach, the authors employ the Jackknife-After-Bootstrap for estimating the variance and an analytical formula for the bias of the parameter in focus. Those estimates are used as inputs in the first-order expansion of the optimal block length expression. The details of this method would require a different approach than the one followed here and for this reason the reader can refer to Lahiri, Furukawa and Lee (2007) or Lahiri (2003). The **NPPI** is given by:

$$\hat{L}_{\mathbf{NPPI}} = \left(\frac{2\hat{C}_2^2}{r\hat{C}_1}\right)^{\frac{1}{r+2}} n^{\frac{1}{r+2}} \tag{15}$$

$$\hat{C}_1 = (nL_*^{-r})n^{2a}\widehat{\mathrm{VAR}} \tag{16}$$

$$\hat{C}_2 = (L_*)n^{a}\widehat{\mathrm{BIAS}} \tag{17}$$

where $r = 1\ and\ a = 0$ for bias and variance estimation, $r = 2\ and\ a = 1/2$ in the case of distribution function. $\widehat{\mathrm{BIAS}}$ and $\widehat{\mathrm{VAR}}$ are the parameter bias and variance consistent (as defined in Lahiri, Furukawa and Lee (2007)) estimates. L_* is a initial block size. For a more comprehensive explanation in the estimation of $\widehat{\mathrm{VAR}}$ the reader might refer to the original paper of Lahiri (2002).

In this monograph, the empirical method for block choice of Hall, Horowitz and Jing (1995), and nonparametric plug-in rule of Lahiri, Furukawa and Lee (2007) are implemented and contrasted.

4.1.1 Algorithms for MBB applied to *GARMA* models

As usual, consider the random variable $Y = \{Y_t(\omega), t \in \mathcal{T},\ \omega \in \Omega\}$, defined in the filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, P)$, where all variables are as previously defined in

section 3. Let $Y_t, t = 1, \dots, n$ denote the realization of the *GARMA* process defined in the filtered probability space where its conditional distribution $f_{Y_t|\mathcal{F}_{t-1}}$ belongs to the same member of the exponential family with linear predictor as a function of its mean ($\eta_t = g(\mu_t)$), as given by equation 2.

Once more, consider $\mathbf{y} = y_1, \dots, y_n$ as a sample from this process. Further, suppose that $n = bL$, where b denotes the number of blocks and L the respective block lengths. Take B as the number of desired bootstrap replicates to perform the Monte Carlo approximation method. **Algorithm 1** describes the steps required to perform the Moving Block Bootstrap for the *GARMA* process Y_t. It can be broken down into six main steps from the creation of the b blocks to the application of the statistic of interest.

Algorithm 1: *GARMA* model MBB

1. Sample b elements, with replacement, from the collection $\mathcal{B}$, where $\mathcal{B} = \{i \in \mathcal{B}: i \in \mathbb{N}, \ \mathrm{i} \leq n - L + 1\}$, to form the set $\mathcal{A} = \{j_1, \dots, j_b\}$;
2. For k in $\mathcal{A}$:
   Compute the blocks of length L starting at the index k;
3. Concatenate the elements obtained in step 2, keeping their indexing order $j_{i's}$, to form the series $\mathbf{y}^* = \mathrm{y}_1^*, \dots, \mathrm{y}_n^*$. This is the first moving block bootstrap replicate.
4. Compute the maximum likelihood estimate of the *GARMA* model parameters evaluated at $\mathbf{y}^*$;
5. Repeat 1-4 B times;
6. Compute the desired statistic, as the estimate of the parameter bias, standard errors, confidence intervals, etc.

In light of the high reliance of the **MBB** procedure into the correct selection of the block length L, a brief adaptation of the algorithm of Hall, Horowitz and Jing (1995) is proposed in order to make it faster and applicable on a multi-parameter setting, resulting in **Algorithm 2**. Speed is relevant when n is large so that we have $m \gg 3$ having to repeat the **MBB** at least m times. To address this issue a sampling scheme is designed to restrict the number of **MBB** runs to 3 on each iteration while trying to cleverly search the parameter space (the possible values of L given $\mathbb{Z}_+^m$). Another drawback of the Hall, Horowitz and Jing (1995) rule is that it is applied to a single parameter, and all the mean squared error estimates are minimized with

respect to one quantity. To overcome this problem a simple solution of a mean optimal block length estimate is adopted.

Algorithm 2: modified Hall, Horowitz and Jing (1995), for $L' \in \mathbb{Z}_{+}^{m}, m > 3$:

1. Choose a value of $m < n$, let $\hat{L}_m$ be the estimate of the optimal block length for the subseries of length m. For each element in $\mathbb{S}$ apply the MBB with the value of L' as defined in the next step.
2. Take L' as: $L' = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, where a is a sample from the quantile 1/3 of $\mathbb{Z}_{+}^{m}$, b from 1/3-2/3 and c 2/3-3/3.
3. Obtain the value $\hat{T}_{L'}^{\mathbb{S}}$ of the statistic of interest and compute $\hat{T}_{L'}^{n}$ and MSE as defined in Section 4.1.
4. For the step $\hat{L}_m = \hat{L}_{m,i} = \underset{i \in L'}{\operatorname{argmin}}\left(\widehat{MSE}\right)$, choose $\hat{L}_m = \sum_{k=1}^{u} \hat{L}_{m,k} \,/u$, where u is the number of parameters in the model (this is useful in a multi-parameter setting, as for each parameter we have a specific $\hat{L}_m$ value).
5. Repeat steps 1-4 j times for j iterations.

If the Hall, Horowitz and Jing (1995) rules and its modified version **Algorithm 2** can generate accurate estimates of L^{Optim} for the **MBB** case, we can safely combine them with **Algorithm 1** and remove arbitrary choices of L in the **MBB** estimation process.

4.2 Bootstrap bias correction and confidence interval estimation

In this section, the parameter bias and confidence interval estimates are defined. Each topic is approached in its designated subsection for the sake of clarity.

4.2.1 Parameter bias and bias corrected estimates

Let $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_n^*)$ be a bootstrapped sample from the original sample of Y_t. If we have B of those samples, and we wish to estimate for example the bias of the parameter ζ (any of the parameters in a *GARMA* model), we should simply compute its value for $i = 1, \dots, B$ and subtract the value of ζ_{MLE} from the mean of the bootstrap estimates, that is:

$$bias\left(\hat{\zeta}_{MLE}\right) = \frac{1}{B}\sum_{i=1}^{B} \hat{\zeta}_{MLE}^{*;\,(i)} - \hat{\zeta}_{MLE} \qquad (18)$$

where $\hat{\zeta}_{MLE}$ denotes the MLE estimate of ζ in the original sample and $\hat{\zeta}_{MLE}^{*;\,(i)}$ in the *i-th* bootstrapped sample. Here, $bias\left(\hat{\zeta}_{MLE}\right)$ is the bootstrap bias estimate of the MLE estimate of ζ. This procedure is the same for all the parameters $\boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{\theta}$ in the respective *GARMA* model.

Thus, we have that $\frac{1}{B}\sum_{i=1}^{B} \hat{\zeta}_{MLE}^{*;\,(i)}$ is the bootstrap estimate of ζ and we could also have the bias corrected estimate of the parameter ζ, *i.e.* $\hat{\zeta}_{BOOT}^{bias\ c.}$, that would be:

$$\hat{\zeta}_{BOOT}^{bias\ c..} = \hat{\zeta}_{MLE} - bias\left(\hat{\zeta}_{MLE}\right) = 2\left(\hat{\zeta}_{MLE}\right) - \frac{1}{B}\sum_{i=1}^{B} \hat{\zeta}_{MLE}^{*;\,(i)}. \qquad (19)$$

4.2.2 Confidence Intervals

With regards to confidence interval estimation for a given parameter (ζ for instance), besides the standard Gaussian asymptotic one, there are some useful bootstrap confidence intervals, such as the *percentile* method, the *bias corrected percentile* method, the *basic* method, among others.

Let $\widehat{\mathcal{H}}$ be the cumulative distribution function of the parametric bootstrap distribution of $\hat{\zeta}^{*}$, so that

$$\widehat{\mathcal{H}}(s) = Prob_{*}\{\hat{\zeta}^{*} \leq s\}. \qquad (20)$$

The *percentile* method:

This is the simplest method to construct a bootstrapped confidence interval for a given parameter and as the Monte Carlo approximation is employed, $\widehat{\mathcal{H}}(s)$ is approximated by:

$$\widehat{\mathcal{H}}(s) \cong \#\{\hat{\zeta}^{*} \leq s\}/B, \qquad (21)$$

where B denotes the number of bootstrap resamples.

Taking $\zeta^{*} \in \left[\widehat{\mathcal{H}}^{-1}(\tau), \widehat{\mathcal{H}}^{-1}(1-\tau)\right]$ gives an approximate $1-2\tau$ confidence interval for ζ, being this the *percentile* method confidence interval.

Chernick and LaBudde (2011 p. 78) provide some remarks regarding this method :

> But asymptotically, the bootstrap samples behave more and more like the subsamples, and the percentile interval estimate does approach the 90% level. Unfortunately, in small to moderate samples for asymmetric or heavy - tailed distributions, the percentile method is not very good and so modifications are required to improve it.

The bias corrected *percentile* method:

In the bias corrected case, take $\zeta^* \in \left[\widehat{\mathcal{H}}^{-1}\left(\Phi(2z_0 - z_\tau)\right), \widehat{\mathcal{H}}^{-1}\left(\Phi(2z_0 + z_\tau)\right)\right]$, where $z_0 \equiv \Phi^{-1}\left(\widehat{\mathcal{H}}\left(\hat{\zeta}^*\right)\right)$ and $z_\tau = \Phi^{-1}(\tau)$. Similarly, $\widehat{\mathcal{H}}(s)$ is assessed by (18). More specifically, through the Monte Carlo approximation to the bootstrap distribution, what is performed is the selection of the 50th percentile of the bootstrap distribution, namely $\widehat{\zeta}^*_{50}$, and the bias correction is taken as $bias\ c. = \hat{\zeta}^* - \widehat{\zeta}^*_{50}$ (Chernick, 2008 p. 60).

The *basic* method:

The basic bootstrap method is similar to the *percentile* method, the difference between then being that in the latter the hypothesis is that the distribution of $\hat{\zeta}^*$ approximates the sampling distribution of $\hat{\zeta}$, while in the former the distribution of $\hat{\zeta}^* - \hat{\zeta}_{MLE} \equiv \hat{\zeta}_{basic}$ approximates the true sampling distribution of $\hat{\zeta}_{MLE} - \zeta$. This method is expected to perform better than the *percentile* when the distribution symmetry condition is not satisfied. The confidence interval construction is analogous to the *percentile* with the replacement of $\hat{\zeta}^*$by $\hat{\zeta}_{basic}$.

Efron and Tibshirani (1986, p. 68) provided a useful table specifying the conditions for each of the previously stated methods to be accurate. For further information regarding those methods, the reader can refer to Efron (1980), Hall (1992), Chernick (2008) and DiCiccio and Efron (1996), just to name a few.

Even though the Gaussian method does not belong to the class of bootstrap confidence intervals we briefly describe it for completeness, as it will be our benchmark for comparison.

The *Gaussian* method:

The Normal case is straightforward and a confidence interval for ζ^* is $\zeta^* \in \left[\hat{\zeta}^* - \hat{\sigma}\Phi^{-1}(\tau), \hat{\zeta}^* + \hat{\sigma}\,\Phi^{-1}(1-\tau)\right]$, where $\Phi^{-1}(\tau)$ denotes the inverse cumulative distribution function of the Gaussian distribution at τ (for a two-sided 95% confidence interval, with τ=2.5% , $\Phi^{-1}(\tau) = -1.96$) and $\hat{\sigma}$ is the MLE estimate of the standard deviation of $\hat{\zeta}$.

The reader acquainted with the theory of bootstrap confidence interval estimation might also acknowledge the use of the bias corrected and acceleration, the bootstrap-t and the ABC (approximate bootstrap confidence interval, which is an analytical version of the bias corrected and acceleration method) intervals. The first requires the estimation of the acceleration parameter, which amounts to evaluate the skewness of the score function and for this reason is not employed. The second requires an estimate of the standard deviation of the parameter of interest in each bootstrap replication, which could be carried out through the delta method or a double bootstrap scheme (increasing the computational cost). The third relies on an analytical approximation as opposed to the Monte Carlo approximation. Therefore, the *Gaussian*, the *basic*, the *percentile* and the bias corrected *percentile* methods are chosen given their easiness of computation and widespread usage.

Usually, for parameter estimates B =100 works fine, as there is little improvement past it, however, for bootstrap confidence interval, a minimal value of B =1000 is required (Efron and Tibshirani, 1986, p. 72).

5 SIMULATION RESULTS

After a previous examination of several initial *GARMA* models (half from the *Poisson* and the other half from the *Gamma*) the final simulation and results were constrained to the scenarios exhibited in

Table 1. As all the proposed models are at most of order one, the respective subscript is omitted (for example, θ_1 is referred simply as θ). This choice of order is made regarding the scope of this work, trying to keep the model structure simpler and easier to interpret. Nevertheless, models with higher order are important and further research is of paramount importance to shed more light into their behavior. Moreover, the initial mean value to generate all models was set to $\mu_{init} = 10$ and in the *Poisson* case the offset parameter to $\alpha = 0.1$.

Table 1: Models and Parameters

		Parameters			
Model	**Family**	ϕ	θ	β	σ
1	*Poisson*	0.15	0.00	2	

2	*Poisson*	0.50	0.00	2	
3	*Poisson*	0.00	0.50	2	
4	*Poisson*	0.00	0.15	2	
5	*Poisson*	0.50	0.10	2	
6	*Gamma*	0.15	0.00	2	1.41
7	*Gamma*	0.50	0.00	2	1.41
8	*Gamma*	0.00	0.50	2	1.41
9	*Gamma*	0.00	0.15	2	1.41
10	*Gamma*	0.50	0.10	2	1.41

As formerly anticipated, this study is concerned with evaluating the performance of the MBB, through a Monte Carlo study, applied to GARMA models with respect to bias and confidence interval estimation. In addition, for the models with a moving average term there was an attempt to implement the Model-Based bootstrap, although, due to the restrict applicability of the methodology and its poor performance no results will be displayed.

Here, a heuristic approach is followed in the selection of the **MBB** block length, where three different values are evaluated for each simulated series length (1000 and 30).

In the case of parameter bias estimation, a graphical evaluation is done, that is, the empirical density (Histogram) of the estimates obtained in the MBB is contrasted to the ones with the bias correction. In this fashion we wish to access whether their empirical distributions look alike and if the introduction of the bias correction term improves the performance of the estimates.

For confidence intervals, the empirical coverage rate of the bootstrap and asymptotic intervals are compared considering a nominal level of 95%, this significance level is chosen regarding its widespread usage in the literature. We understand that the bootstrap performance might be related to the chosen significance level and consider this a fruitful field for future research. In the simulation study a total of 1000 Monte Carlo repetitions and 1000 bootstrap replications will be performed. All the codes are built in R and are presented in Appendix III: R CODE.

As the discussion of all the scenarios described in Table 1 would be a little cumbersome, only the results of some models are presented here. But the performance for all models can be found at the Appendix I: Tables and Appendix II: Figures.

We will take Model 1 as example. Figure 5 presents the results for n=1000. There are three plots in the same figure and their interpretation is the same, the difference is their block length, being the first for the length of 20, the second 50 and the last for 100. For instance,

consider the second graph in this figure; the red area depicts the histogram of the density of the original 1000 MLE estimates for the parameter ϕ in Model 1, while the blue area shows the density of the respective bias corrected estimates. In this case there is almost a complete overlapping between then, meaning that the distributions are approximately the same. This distinction is made clear because in some cases provided in the Appendix there is no overlapping at all. For this reason, three vertical lines are added to the plot in order to assist the visual diagnosis. The black line represents the parameter true value (on which the estimates distributions are expected to be centered), and the red and blue the respective original and bias corrected mean MLE parameter estimates. In the case in analysis, increasing the block length improves the point estimates of ϕ, but does not affect the estimates of β. Besides, the bias corrected estimates present the same performance as the case without the correction.

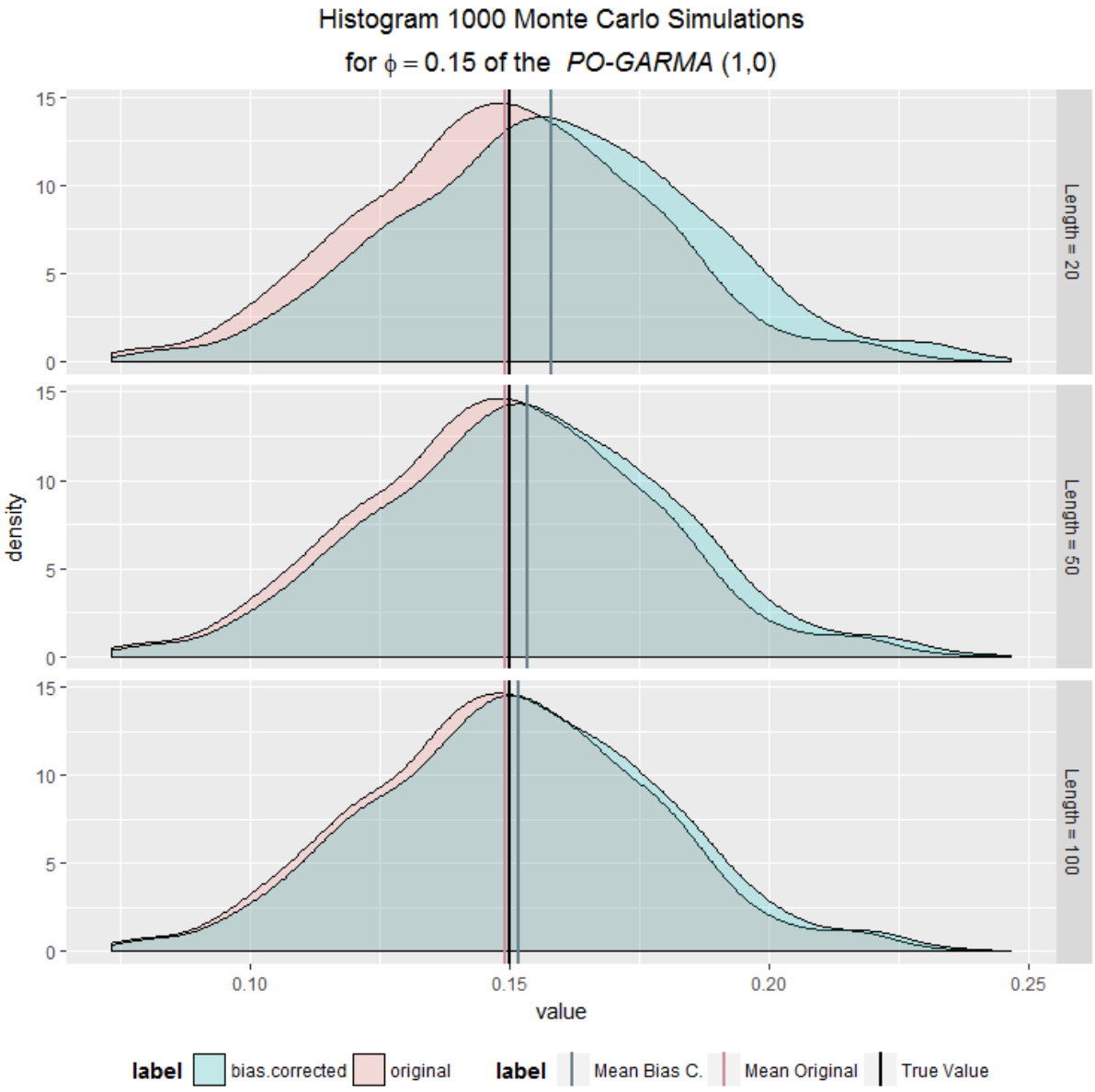


Figure 5: Model 1 – $\hat{\phi}$ original and bias corrected empirical distribution for 1000 simulations, series of length 1000

Small sample properties of the estimates are also evaluated, with a Monte Carlo simulation performed for a series of length 30 (see Figure 6). Results for the smaller series are worse than the larger one, as expected, with both original and bias corrected empirical distribution showing a larger variability. Moreover, we see that the mean bias corrected estimates are closer (in Euclidian distance) to the true value of ϕ compared to their counterpart without the correction, and it improves as the block length increases.

An example with a larger value of the parameter is presented in Figure 7 for Model 3. We have perceived that the results are worse when we increase the parameter values, and this is probably due to the fact that the MBB was built for stationary series. Thus, a slight departure from this assumption can imply in estimates farther from the real values. Nevertheless, there

are some cases when the bias correction can improve the results, as shows the graphs in Figure 7. We see that the bias correction brings the estimates of θ closer to the real one when we decrease the block length.

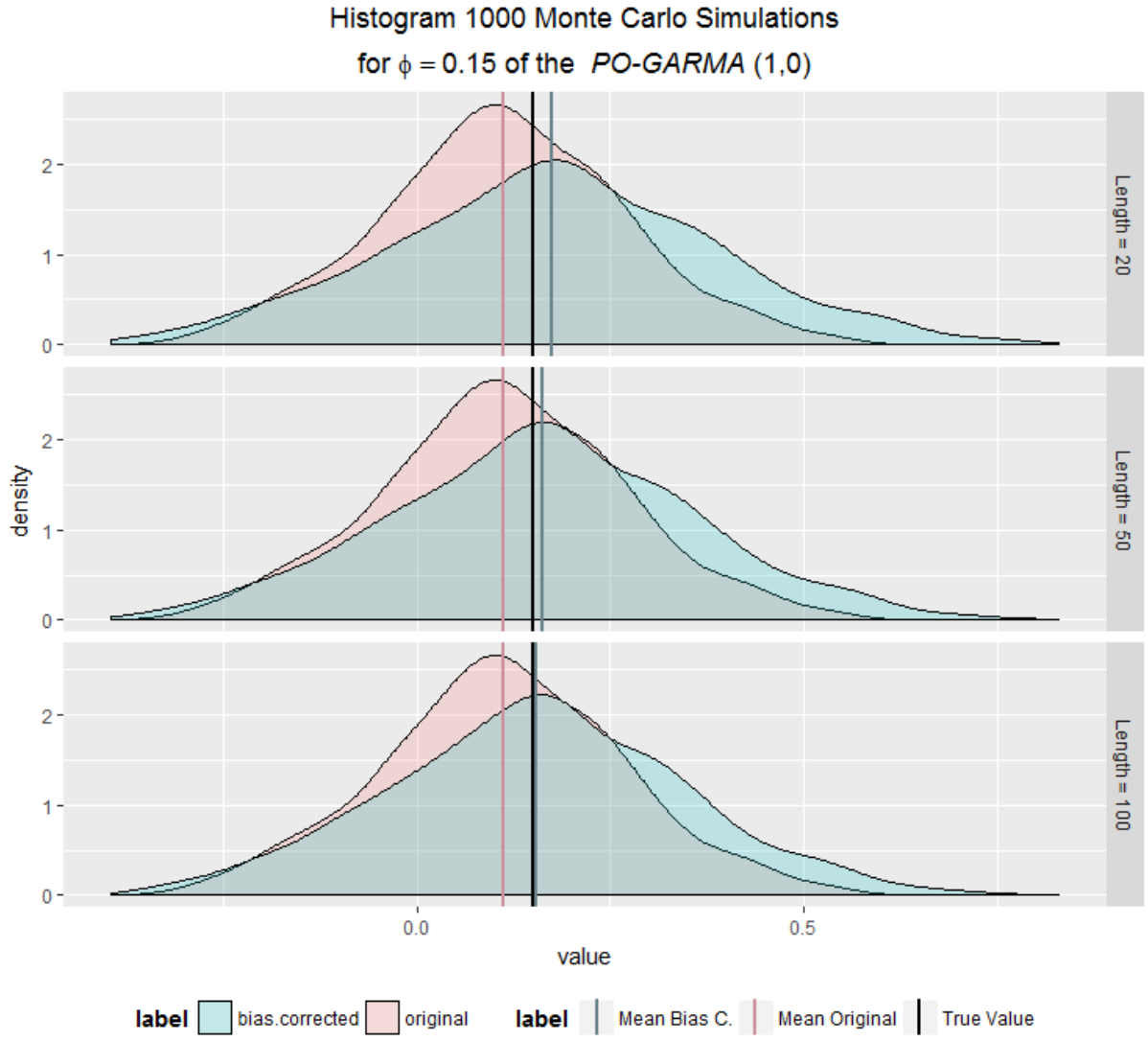


Figure 6: Model 1 - $\hat{\phi}$ original and bias corrected empirical distribution for 1000 simulations, series of length 30

In general, increasing the block length improves the performance of the estimates, except for MA models in Poisson GARMA.

In spite of the superiority of the bias corrected parameter estimate detected in Model 3, in some models this might not be the case. For example, in Model 5, for ϕ, the original estimates are better than the corrected ones. Likewise, in Model 9 for β, no clear distinction can be made between them (see Figure 23 and Figure 27 in the Appendix II: Figures).

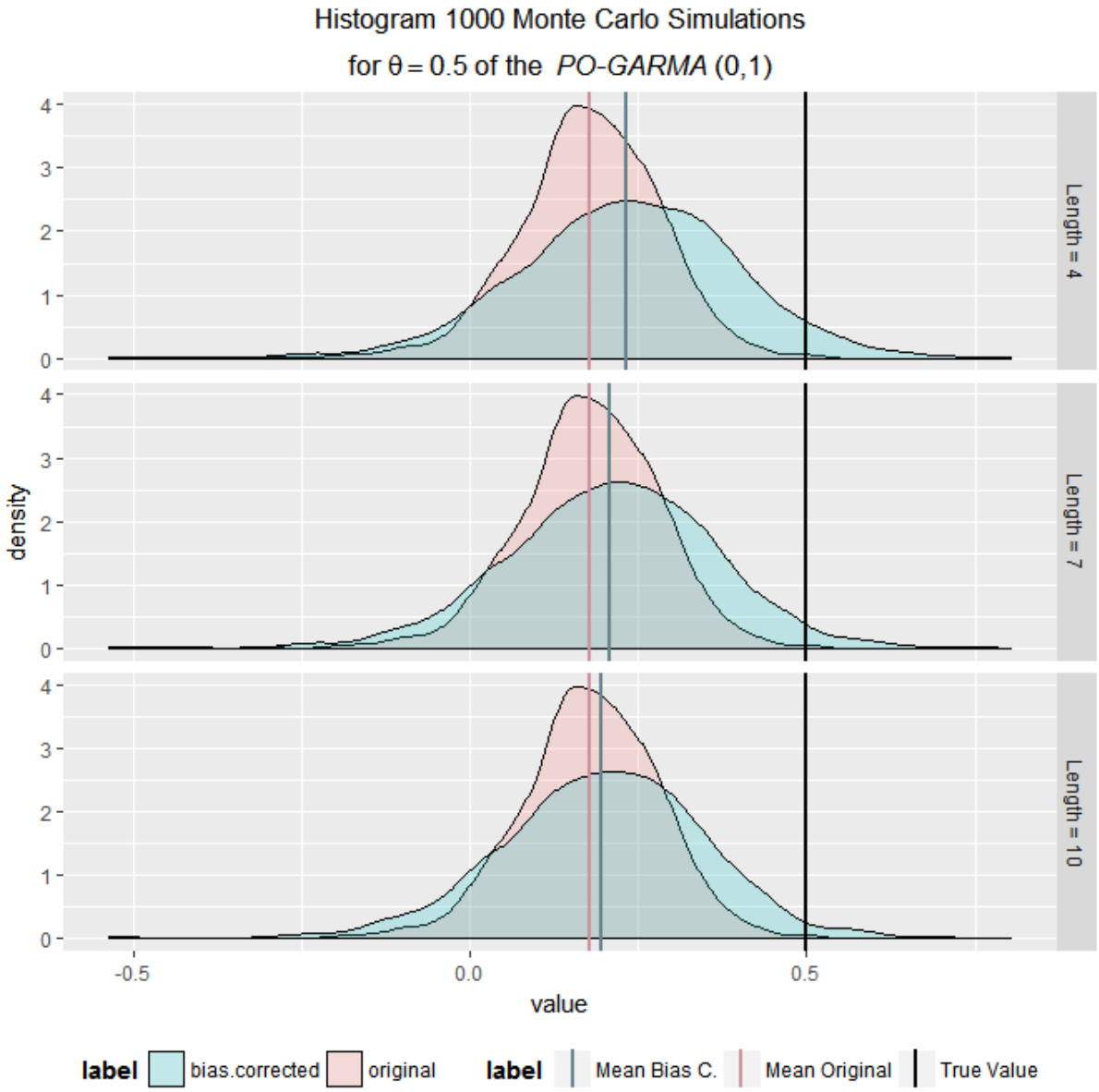


Figure 7: Model 3 – θ original and bias corrected empirical distribution for 1000 simulations, series of length 30

With respect to coverage rates, some selected models are analyzed and a classification is proposed to simplify the exposition and understanding, but the tables with the data for all the models can be found at Table 5 to 8 at the Appendix I: Tables.

Table 2: Bootstrapped confidence intervals coverage, 1000 sim., 1000 boot. resamples, series of length 1000

		Model 3: Poisson-GARMA (0,1)			
		β=2; θ=0,5			
length	**parameter**	**norm**	**bias c.**	**perc.**	**basic**
20	**β**	93.7%	93.3%	93.9% *	93.2%
	θ	79.6%	93.3% *	30.1%	92.8% *
50	**β**	93.7%	92.7%	93.3%	92.9%
	θ	79.6%	85.7% *	72.0%	86.3% *
100	**β**	93.7%	91.2%	92.6%	91.7%
	θ	79.6%	75.6%	79.1%	75.8%

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal

Table 2, displays the coverage rate for the confidence intervals of 1000 Monte Carlo simulations from a *Poisson-GARMA* (0,1) model, with $\beta = 2$ and $\theta = 0.5$. The column labels have the following meaning: **length** denotes de block length from the moving block bootstrap; **norm** is the Gaussian confidence interval and it reads normal confidence interval without bias correction; **bias.c** indicates the normal confidence interval with the bias correction term; **perc.** stands for the percentile confidence interval and **basic** designates the basic confidence interval. All previous intervals are as defined in Section 4.2..

For this model, the asymptotic interval had an erratic behavior, as displayed in Table 2. In fact, we see that the asymptotic interval fails for the parameter θ, with a 79.6% coverage rate, indeed a poor performance. On the other hand, the bias corrected normal interval and the percentile method do a good job for some block lengths, where the coverage rate is closer to 95%. Moreover, for some values of block length the percentile method had a better performance regarding β.

However, the results from Model 3 do not tell the whole picture. In some instances, the MBB failed at all for some parameters and for others it worked better for a specific block length. For models 2, 6 and 7 there is a complete failure, that is, no single block length and confidence interval had a higher coverage rate than the asymptotic one. The outcome for models 1, 4, 5, 8, 9 and 10 are analogous to the case of β in model three. That is the case because there is no failure in the asymptotic interval; it only has a poorer performance than the others confidence intervals in terms of coverage rates do.

It is worth mentioning that this is a partial failure, as for models with coefficients of 0.15 (models: 1, 4, 6 and 9) their performance is superior to the one's with 0.5 (models: 2, 3,5,7,8 and 10). Actually, their coverage rates are above 90% (except one case in model 9, 89.5%) for all block lengths and confidence intervals and for specific combinations, this value gets arbitrarily closer to the target of 95%. Thus, this is a strong evidence in favor of the hypothesis of using the MBB for parameter bias and confidence interval estimation. The poor performance of the other models was already expected, as a value of ϕ or θ of 0.5 (closer to 1) is usually associated with non-stationarity and non-invertibility of the series. However, even in the latter cases, we do observe some models where there is a combination of block length and confidence interval that leads to values close to the 95% threshold (for instance, consider model 5 with block length of 50 for the percentile ci).

Additionally, the relative better performance of the MBB confidence interval estimation over the asymptotic one is associated with the presence of a moving average term in the model. The only pure autoregressive model that had higher coverage rates than the Gaussian was model 1, though, the asymptotic interval clearly is better, as it is closer to the desired 95% coverage rate for a majority of block lengths. Pure moving average model 3 and 8 and the ARMA models 5 and 10 do exhibit higher coverage rates than the reference, whereas in some instances the asymptotic one is favored.

Table 3: Bootstrapped confidence intervals coverage, 1000 sim., 1000 boot. resamples, series of length 30

		Model 3: Poisson-GARMA (0,1)			
		β=2; θ=0,5			
length	**parameter**	**norm**	**bias.c**	**perc.**	**basic**
4	**β**	85.1%	83.9%	89.1% *	85.3% *
	θ	11.4%	48.5% *	6,0%	54.5% *
7	**β**	85.1%	81.6%	86% *	82.9%
	θ	11.4%	40.9% *	12.4% *	47.7% *
10	**β**	85.1%	76.9%	82.4%	77,0%
	θ	11.4%	36.6% *	14.6% *	42.1% *

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal

This behavior is observable in small samples. Table 7 and Table 8 at the Appendix I: Tables exhibit the coverage rate for the same models with a series of length 30 and block length of 4,7 and 10. Similarly, model 3 seems to fail for the asymptotic case and the bootstrapped results can improve the estimation of the parameter β, as we can see in Table 3. However, for

the parameter θ the bootstrapped confidence intervals show little improvement over the Gaussian, with low coverage rates. In general, for models with a moving average term, there might be a benefit from using the bootstrap ci for some models and parameters. Notwithstanding, for other models (e.g. model 1) the asymptotic ci coverage rate is closer to the expected and superior to the bootstrapped ones. In contrast to the large sample case, in small samples, models 4, 6 and 9 do exhibit a combination of block length and confidence intervals that are superior to the Gaussian counterpart.

Generally, the results support the thesis that the moving block bootstrap, with a given block length, can improve the results for confidence interval estimation, when contrasted to the Gaussian asymptotic ones, in the specific case of failure of the asymptotic in the simulated *GARMA* models. In small samples, the benefit from using block-resampling methods is superior to the asymptotic Gaussian, as an increased number of models display higher coverage rates than the reference. Again, models with parameters with values closer to unit do exhibit coverage rates inferior to the ones with lower values (0.15 for instance), reinforcing that non-stationarity and non-invertibility is one of the main concerns when dealing with bootstrap procedures in *GARMA* models. Nevertheless, care must be taken in choosing the appropriate block length ensuring the desired results. Perhaps a plug-in estimate of it might lead to a less heuristic and empirical work and even better results.

The general guideline regarding confidence interval, their coverage rates and *GARMA* models can be divided in two categories: large and small samples. In the case of a large time series (e.g. n=1000) the reader should favor the asymptotic approximate confidence interval. On the other hand, for small time series (e.g. n=30) the reader is advised against the usage of the basic confidence interval and should focus on using the percentile or bias corrected percentile ci's.

5.1 Optimal block length

In this section, the optimal block length is evaluated and, as the examples provided bellow show, $L_{G.}^{Optim}$ might not be a plausible measure. First, optimality for the model with $n = 30$ is assessed, with 100 replications for each block length from 1 to 20, and a summary of the L^{Optim} values for all model parameters is presented in Table 4. The results support the hypothesis that $L_{G.}^{Optim}$ is not a good measure, as each parameter optimal values differ, for instance, consider model 2, where $L^{Optim}(\beta) = 10$ and $L^{Optim}(\phi) = 15$.

Another interesting perspective can be seen in Figure 8 where the boxplots for 100 MBB simulations of Model 2 for every block length are computed, alongside with the mean values (blue dots) and optimal block length (orange dots, $L^{Optim}(\beta) = 10$ and $L^{Optim}(\phi) = 15$). Small block lengths leads to higher dispersion and higher block lengths to a smaller dispersion of bootstrap resample estimates. For ϕ, the mean MBB resample parameter values can roughly be seen as a monotone increasing function of L.

Table 4: Optimal block length for all models and parameters

	Parameter			
Model	β	ϕ	θ	σ
1	8	8	-	-
2	10	15	-	-
3	1	-	1	-
4	1	-	1	-
5	19	15	20	-
6	15	5	-	1
7	18	9	-	1
8	15	-	15	1
9	15	-	11	1
10	16	12	13	5

Overall, the estimation of β seems more stable (with respect to L) than the other model parameters. This means that one might eventually neglect the effect of the chosen block length on estimating β, and consequently focus only on the remaining parameters. Thus, $L_{G.}^{Optim}$, might be redefined in $\mathcal{K}\backslash\boldsymbol{\beta}$. In this fashion, one could also put in second place the block length effect for the estimation of σ. For this reason, it is safe to restrict the attention to $\boldsymbol{\phi}'$ and $\boldsymbol{\theta}'$.

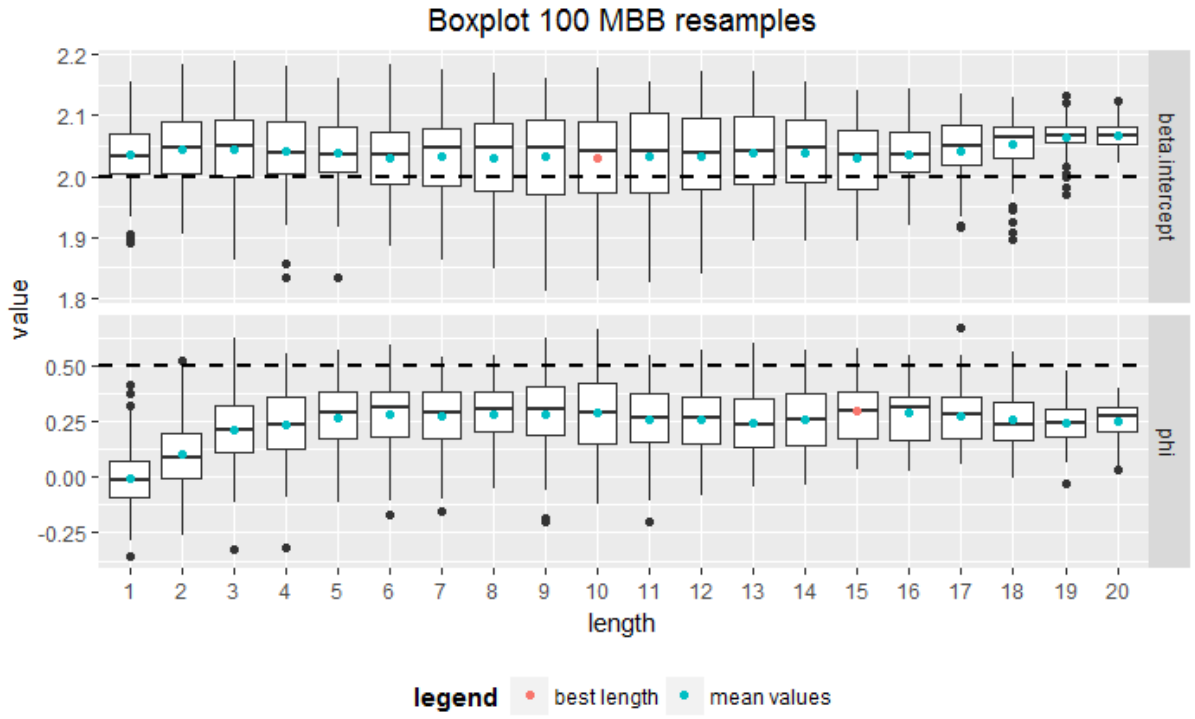


Figure 8: Model 2 optimal block length, series of length 30, n=30, β=2, ϕ=0.5, boxplot for 100MBB simulations

For a simulated series, it is possible to know the optimal block length for a given parameter (though for long series it might take too long to compute it), however, with real datasets, some heuristic or algorithm must be employed. In this fashion, in Figure 9 the method proposed by Hall, Horowitz and Jing (1995) (henceforth **HHJ**) is iterated (as suggested by the authors) 100 times and the frequencies of the optimal L on each step are tabulated for models with an AR component. The optimization is taken with respect to ϕ. The results of the algorithm do not always agree with the optimal behavior depicted in Table 4. The results can be grouped into two cases: oscillatory behavior and convergence, the former indicates that the algorithm oscillates between a group of values, while the in the latter after a number of iterations we have a convergence of the algorithm to a unique solution. It was established that $L^{Optim}_{Model\ 1}(\phi) = 8$, but the algorithm oscillates between 2 and 3, though no step points to the true value. The same reasoning of oscillatory can be applied for Model 6 and 10. Still in the first case we have that $L^{Optim}_{Model\ 2}(\phi) = 15$ and the **HHJ** steps have high frequency at $L = \{7,8,9\}$. In the second case, the **HHJ** converges fast (though not to the true optimal value) as in Model 5 that converged to an optimal block length of 19 and in Model 7 to 17.

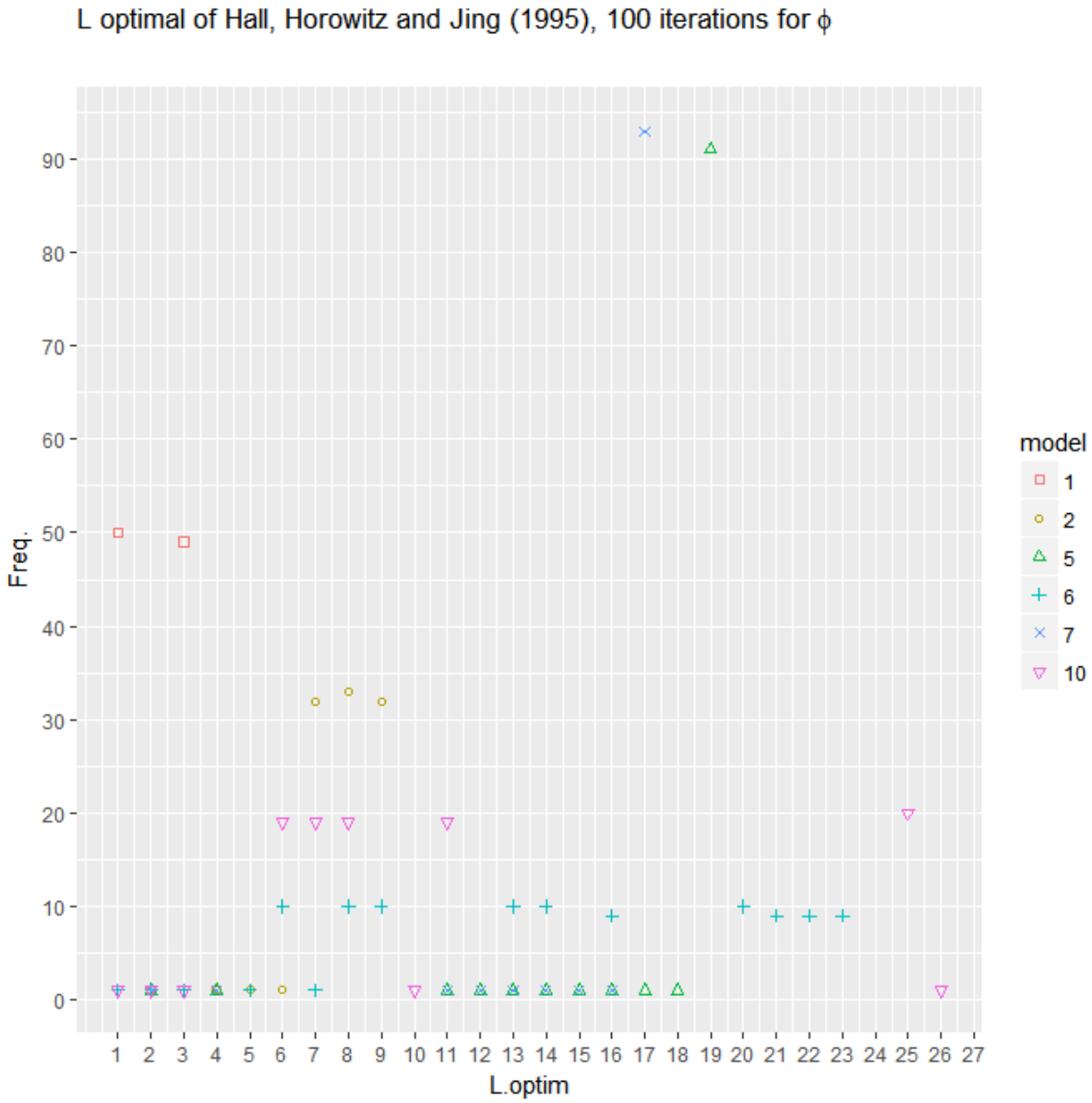


Figure 9: ϕ optimal block length under Hall, Horowitz and Jing (1995) algorithm, 100 simulations block length frequencies

The previous results show that the **HHJ** did not have a satisfactory result. Overall, this algorithm does not seem appropriate for selecting the optimal block length for the MBB in the *Poisson and Gamma GARMA* models context. It is worth noting that the implementation here was with respect to ϕ and to bias and variance estimation, thus there could be a different value of L for estimating other model parameters and confidence intervals. Additionally, the results of the **HHJ** algorithm rely on the initial value and random seed, so there might be no unicity on the chosen block length values. This multiplicity of block length possible values increases the parameter space and favor the heuristic approach. Also, this hypothesis is corroborated if we take into account the approximate monotone behavior of the block length distribution (as in Figure 8).

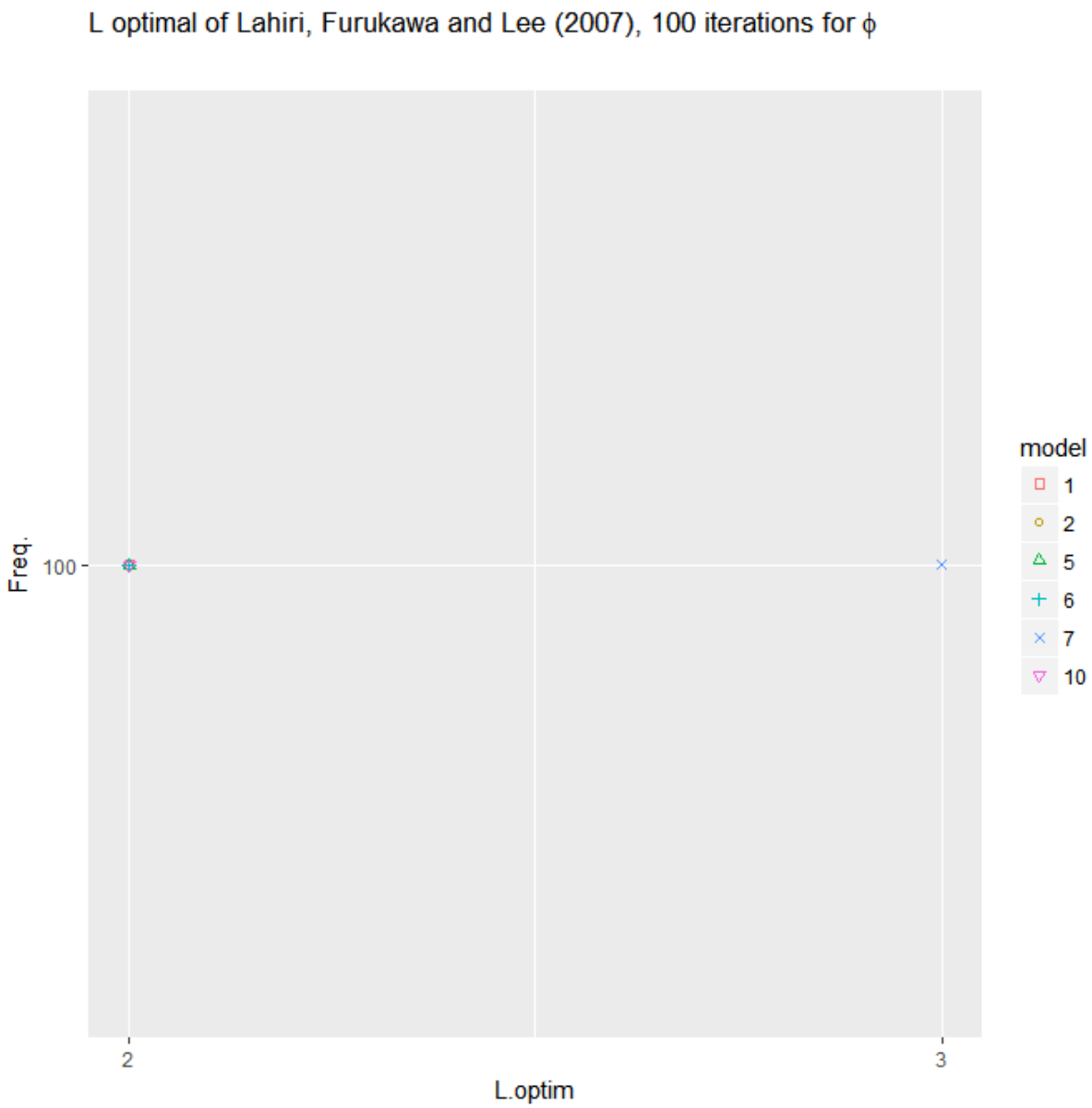


Figure 10: ϕ optimal block length under Lahiri, Furukawa and Lee (2007) algorithm, 100 simulations block length frequencies

Repeating the same analysis for the **NPPI** algorithm, after 100 iterations we have that $L = 2$ in all instances. All models had a constant optimal block length estimate (with respect to the iteration span). It is clear that the **NPPI** algorithm does not work properly in these models, as we know from Table 4, above, that the true optimal block length is $L_{Model\ i}^{Optim}(\phi) = \{25,\ 25, 30,\ 5,\ 25,\ 12\}, for\ i = 1, 2, 5, 6, 7, 10.$

6 REAL DATA ANALYSIS

In this Section two examples are provided, one for the *Poisson* and the other for the *Gamma – GARMA* models.

6.1 The *Poisson – GARMA* case

A dataset with the number of monthly bankruptcy filings in the USA from the UCLA-LoPucki Bankruptcy Research Database was used as an example of modelling a *Poisson* time series. It encompasses public companies with Annual Report reporting assets worth 100 million of U$ dollars or more and can be downloaded at Federal Reserve Bank of St. Louis website (https://www.stlouisfed.org/). The analysis is restricted to the period of January of 1980 to December of 1999 and a plot of the series is provided in Figure 11.

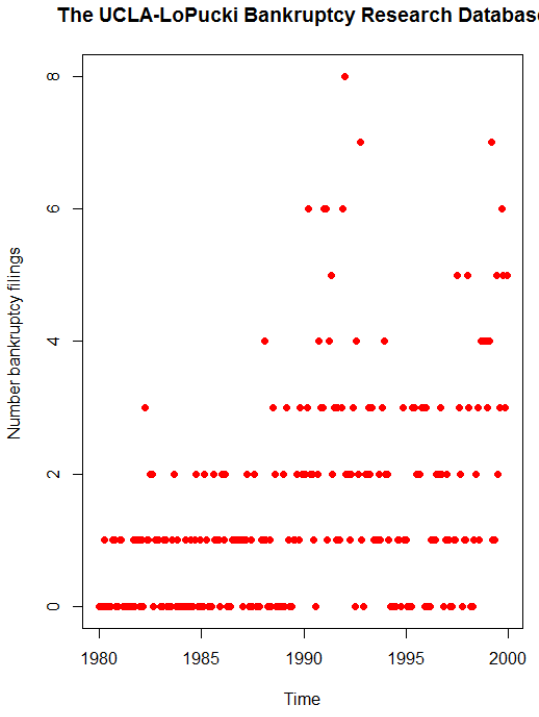


Figure 11: Bankruptcy Series plot

This topic is relevant as the number of bankruptcy filings is a key economic outlook variable, helping to diagnosis the current status of the economy and also working as a lagging indicator in the business cycle. Figure 11 displays the number of bankruptcy filings by month over the years, there is an upward increasing trend. As the intention is only to demonstrate the

application of the *GARMA* model, no more economic digressions will be made. Thus, a plot of the autocorrelation function (ACF) and the partial autocorrelation function (PACF) is required to assist in the ARMA order detection. This is a standard procedure in the Gaussian framework, the interest reader might refer to Penã, Tiao and Tsay (2001).

Figure 12 shows the ACF and PACF of the bankruptcy filings time series. The high first order autocorrelation value with its slowly decaying pattern, associated with the high first order partial autocorrelation and zero autocorrelation from higher order lags indicates a AR(1) or even an AR(2) model.

After identifying the models and proceeding with the estimation of the *Poisson-GARMA*(1,0) model, the parameter estimates are: $\hat{\beta} = 0.6132$ and $\hat{\phi} = 0.3396$; both statistically significant at the level of significance of 1%. Moreover, the AIC statistic for this model is 758.323.

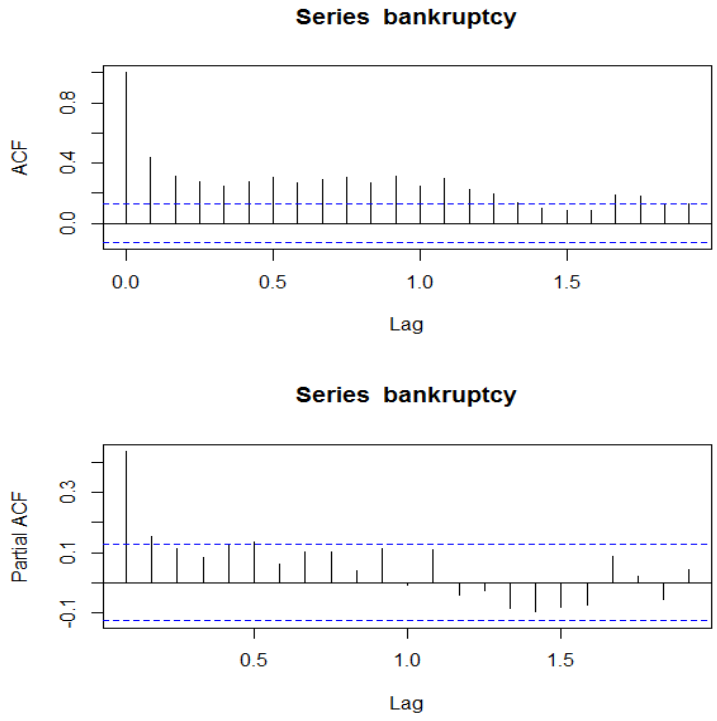


Figure 12: ACF and PACF plots of the Bankruptcy Filings Series

In Figure 13 there are some plots for the residual diagnostic. The Residual X Index plot shows no discernible pattern of the residuals. In addition, the Normal Q-Q Plot highlights the approximately Gaussian behavior of the residuals, except for the distribution tails. Those facts support the hypothesis of no model misspecification.

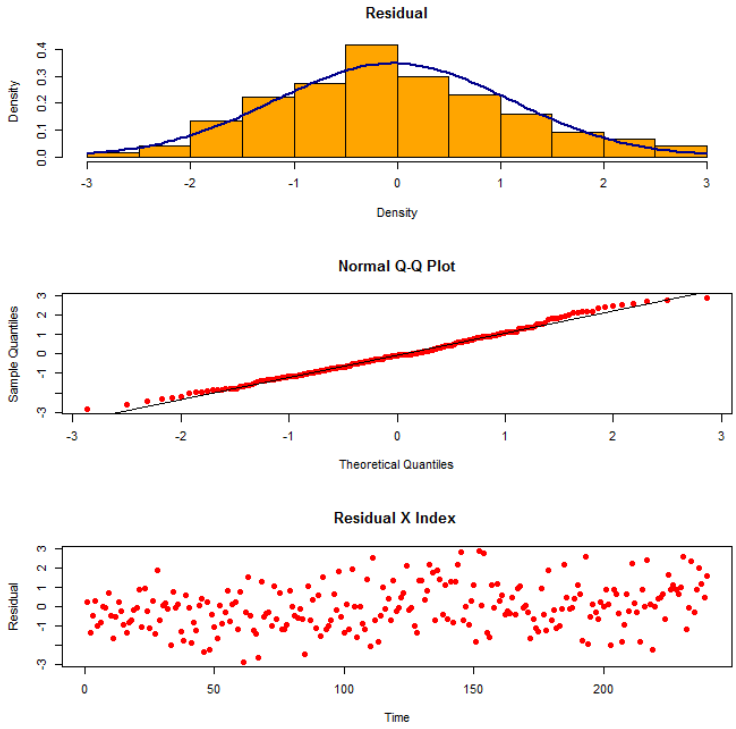


Figure 13: Residual Analysis of the Poisson-GARMA(1,0) model

Alternatively, we can also attempt to estimate a *Poisson-GARMA*(2,0) model. Here we have that $\hat{\beta} = 0.7534$, $\hat{\phi}_1 = 0.2612$ and $\hat{\phi}_2 = 0.1790$, all the parameters are statically different from zero at the significance level of 1%. The AIC off the model is 735.597, which indicates that model 2 is preferable over model 1, as it minimizes the information criteria. As usual, in Figure 14, the residual analysis shows that the data fits the model accurately, with no severe deviation from the Gaussian hypothesis in the Normal Q-Q plot and also there is no clear pattern in the residual X index plot.

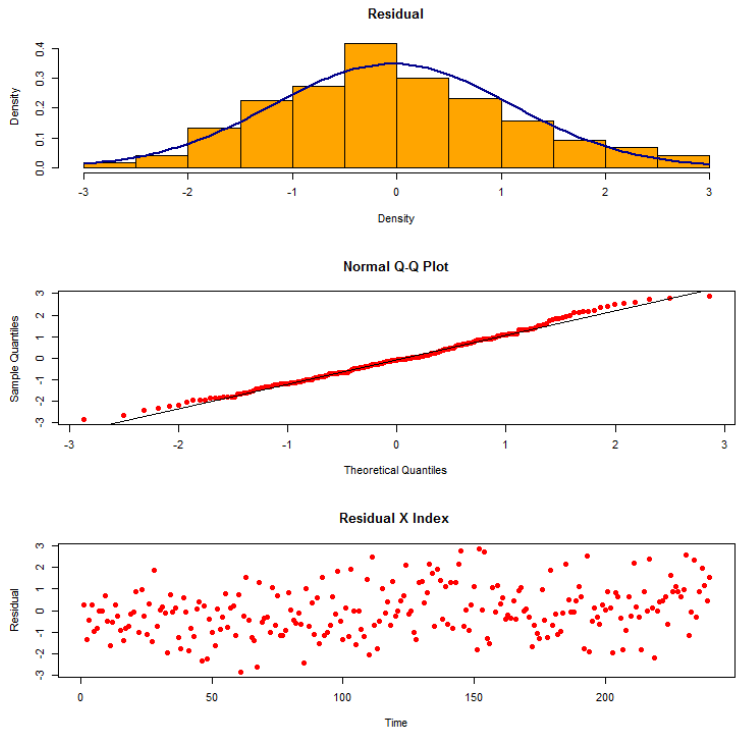


Figure 14: Residual Analysis of the Poisson-GARMA(2,0) model

In terms of confidence intervals, Figure 15 exhibit the mean value of 1000 bootstrap resamples for model 2 alongside with the error bars for the 95% confidence interval. Both the Normal asymptotic and the one with the bias correction term are displayed. Note that the MBB has been computed for a grid of 16 possible block lengths, with values ranging from 5 to 160, with a constant difference of 10. In addition, the ci's with the bias correction term are much wider than the Gaussian ones, though, the mean of the bootstrap resamples belongs to all of them in the former, whilst in the latter, in some instances, the mean of the bootstrap resamples does not belong to the interval.

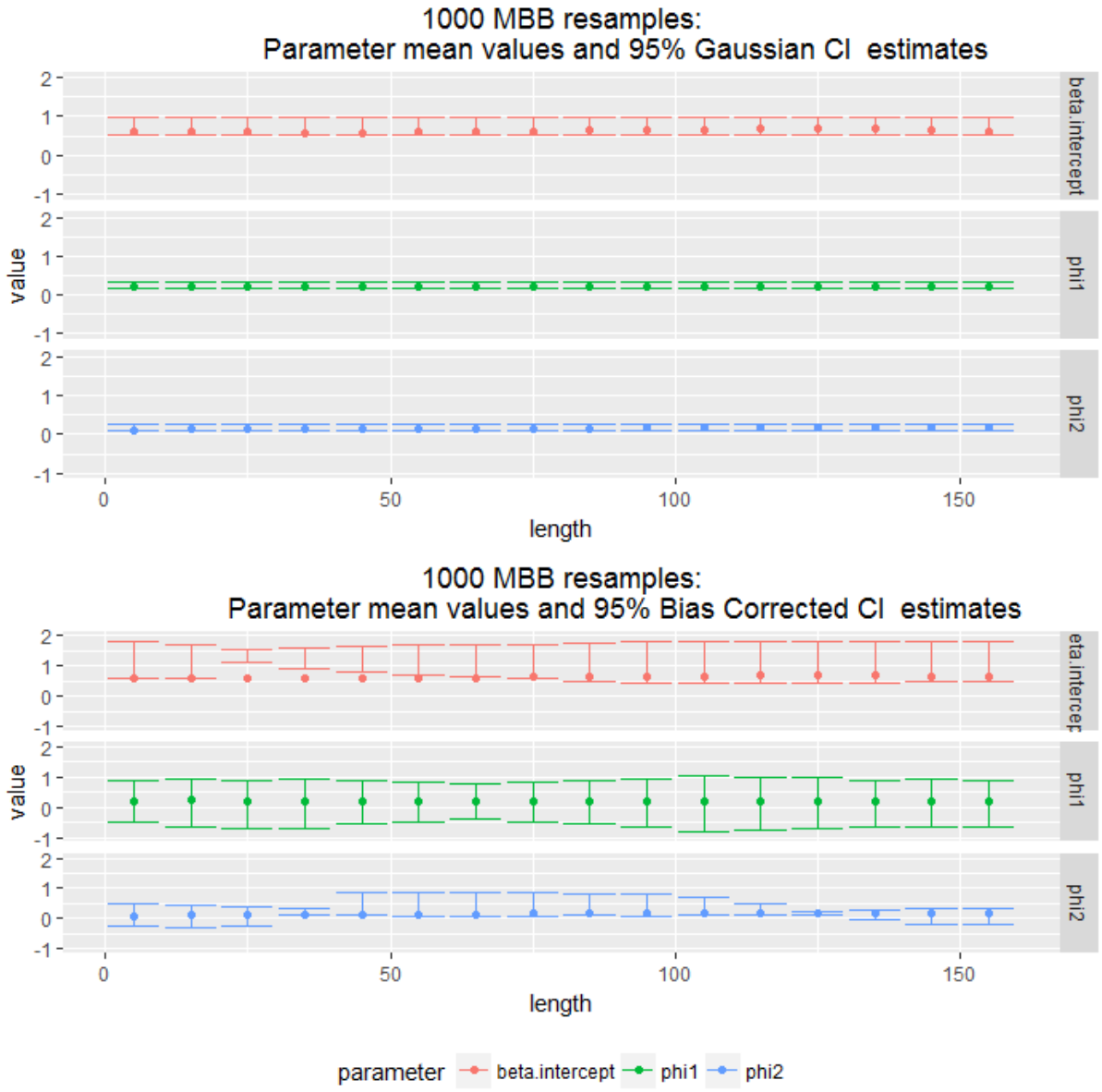


Figure 15: MBB 95% Confidence Intervals: Normal and Normal with bias correction term for the Poisson-*GARMA*(2,0) model, from 1000 resamples

As there are 240 observations in the time series, computing the MBB for all possible block lengths can take a considerable amount of time. Owing to this, the analysis will be restricted to **Algorithm 2.** In the case of bias/variance and confidence intervals (two sided distributions) estimates, **Algorithm 2**, for an initial value of $m = 30$, in 10 steps, all resulted in a value of $L = 13$ ($R = 100$). Besides, monotone convergence for high values, close to the number of observations, or in the opposite case, for small values, is not the best possible outcome. The former results in low bias and increased variance, while the latter decreases the variance and increase the bias of the estimates. What we wish to achieve is an appropriate balance between bias and variance in the bootstrap resamples.

6.2 The *Gamma – GARMA* / GARCH case

For the final GARCH model, 180 observations from the series of log-returns (difference of log prices) of Brasil Pharma S.A. (ticker BPHA3.SA), a pharmaceutical company, were used, with dates ranging from 2017-05-30 to 2018-02-22. In order to check whether there are ARCH effects or not in the time series two tests were performed, the ARCH test and the Ljung-Box statistic to the squared log-returns (or volatility if we assume a zero mean). Both are standard procedures and can be found at Tsay (2002). Both tests were performed using 1, 4, 8, 12 and 24 lags, with the series length equal to the last (chronological order) 30,60,90,180,360,504 business days. At the level of confidence of 10%, for the ARCH test, only at length 30, lags 8,12,24, length 60, lag 24 and length 360, lags 4,12,24 we do not reject the null hypothesis of no ARCH effects. By the same token, for the Ljung-Box test, only for the length 360 and lags 4,8,24 that the null of no autocorrelation is not rejected (actually the test is for all joint lags, and that is why six different values were used).

Thus, knowing that in the majority of combination of series length and test lags we do not reject the presence of ARCH effects we can proceed in the estimation of the GARCH model. A useful procedure in selecting the process order is a graphical inspection of the Autocorrelation Function and the Partial Autocorrelation Function of the squared log-returns

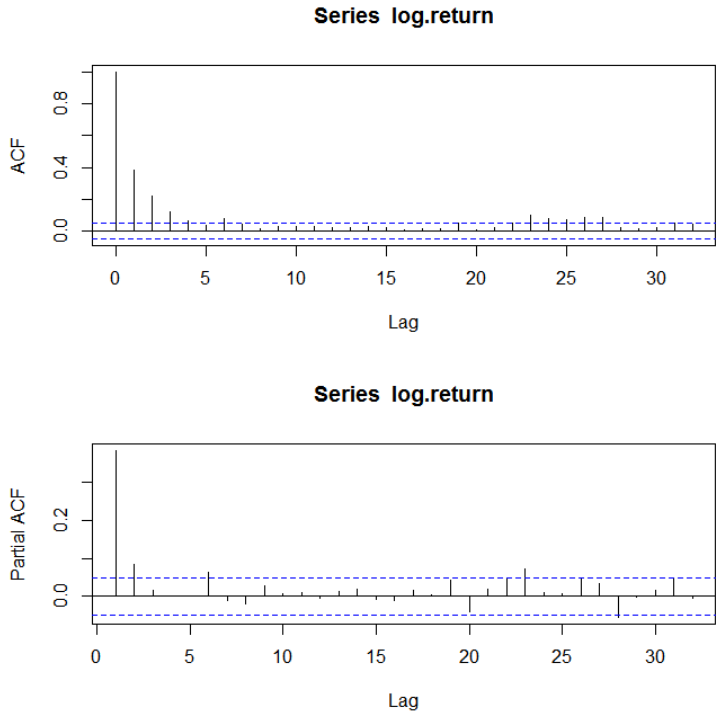


Figure 16: ACF and PACF of the BPHA3 squared log-returns

One of the possible models to be estimated is a *Gamma GARMA/GARCH* (1, 0) process, as we have a decreasing behavior in the ACF and another one at lag 1 of the PACF, see Figure 16. The significative spike at lag 2 might indicate an AR(2) process, though, due to the widespread usage of the GARCH(1,0) we will restrict the attention to the aforementioned model. The parameters estimated values are: $\hat{\beta} = 1.9886$, $\hat{\phi} = 0.0247$ and $\hat{\sigma} = 1.2959$, with all of them significant at the level of 1%. After 1000 replications of the estimation process, the mean estimates are $\bar{\hat{\beta}} = 1.9887$, $\bar{\hat{\phi}} = 0.0247$ and $\bar{\hat{\sigma}} = 1.2960$, with no differences in maximum and minimum with four decimal points.

A first model diagnostic is the residual analysis, in Figure 17. We see that some observations in the tail of the distribution in the Normal Q-Q Plot do not behave like a Gaussian distribution. From the theory of financial time series, we know that this is a possible behavior that can be generated by abnormal returns. Additionally, returns usually have higher probability at the tails of its distribution when contrasted to a Normal distribution and even GARCH models with normal errors might fail to capture the series true data generating process. Modern GARCH processes incorporate other distributions other than the Normal, such as the Student *t* distribution, the generalized error distribution and the generalized hyperbolic distribution.

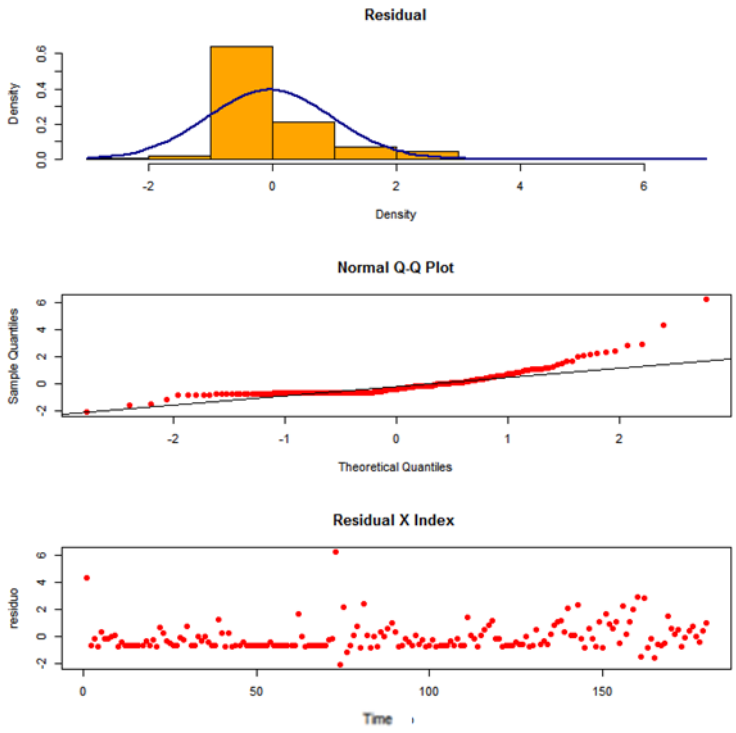


Figure 17: Residual Analysis of the Gamma-GARMA/GARCH (1,0) model

Furthermore, another useful estimation diagnostic is the computation of the ARCH and Ljung-Box tests on the model residuals to check if there are any ARCH effects left. For lags

1,4,8,12 and 24 all tests do not reject the null hypothesis of no ARCH effects / autocorrelation statistically different from zero. This simple procedure is a strong evidence in favor of the estimated model adequacy (at least in terms of ARCH effects that was the main concern for estimating a conditional volatility model).

For **Algorithm 2,** bias/variance and confidence interval estimation, for an initial value of $m = 10$, $R = 100$, in 10 steps, there was a convergence for $L = 12$ after one iteration. In Figure 18 are the charts of the Gaussian and bias corrected ci estimates using a grid of values for L. Both mean estimated values are contained in the respective ci's except that the bias corrected is wider than the Normal. We see that the estimates are relatively stable over the set of chosen block lengths and chosen confidence interval type.

After examining both real data examples we achieve the same conclusion, that is, **Algorithm 2** is a powerful tool in assisting in choosing the block length for the MBB algorithm, yet it is also prone to non-unicity. This drawback reinforces the importance of the heuristic approach, by which we should always estimate the MBB over a grid of values to check the model estimates variability.

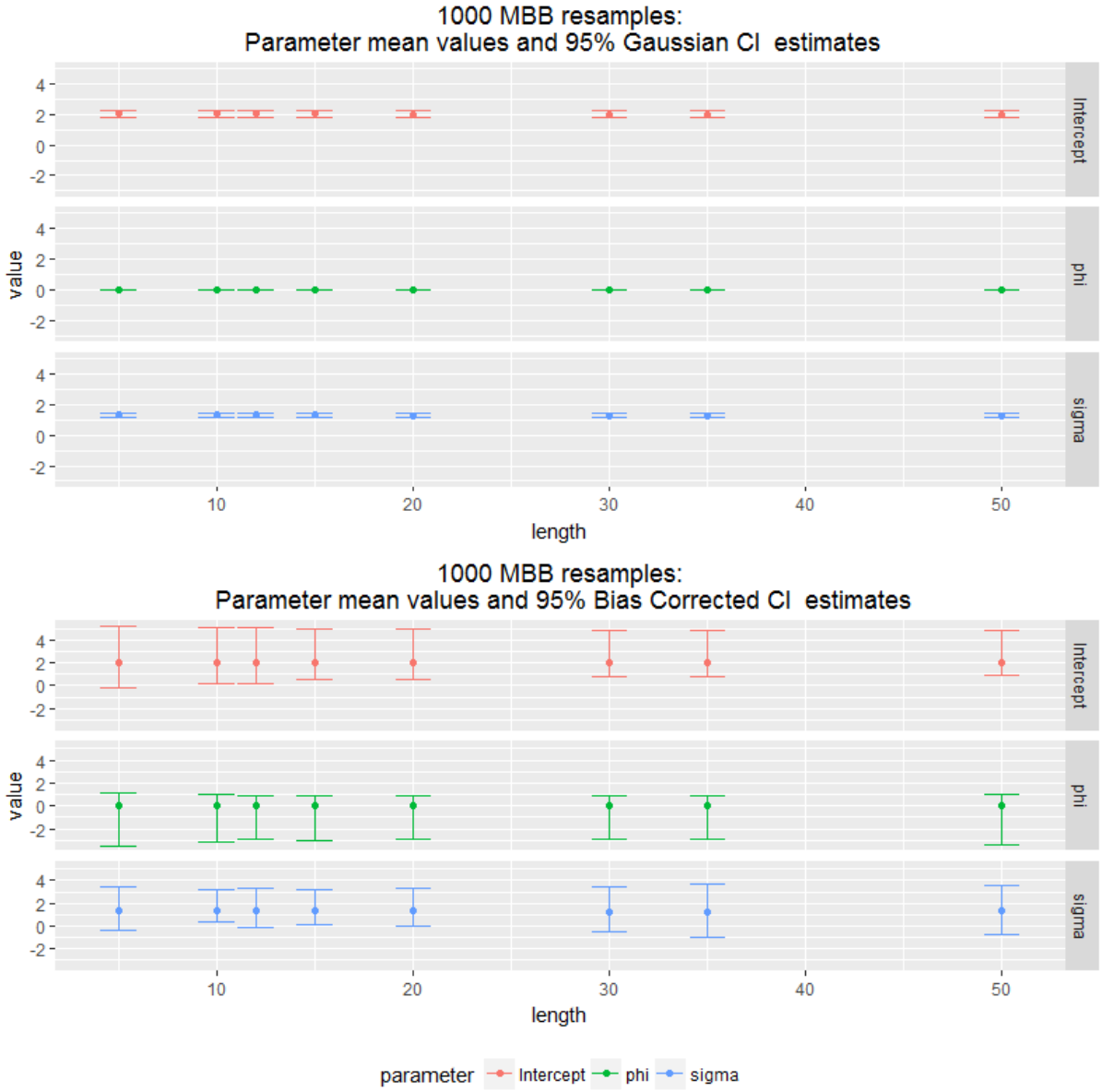


Figure 18: MBB 95% Confidence Intervals: Normal and Normal with bias correction term Gamma-GARMA/GARCH model, from 1000 resamples

7 CONCLUSION

In conclusion, bootstrap application in the context of the *GARMA* model is nonstandard and as such, suitable modifications must be employed. The Moving Block Bootstrap seems like a reasonable solution to tackle this problem, and the bias corrected parameter estimates and confidence intervals might work in cases where the Gaussian counterpart fail. This phenomena happens even in large samples from selected *Poisson* and *Gamma GARMA* process, where the coverage rates of the MBB surpass the asymptotic ones. Remarkably, in small samples, from size of 30, these properties hold.

Furthermore, the performance of the MBB is related to the nature of the terms present in the model, as models containing a moving average term showed improvement over their references. In the other hand, pure autoregressive models generally performed poorer than the benchmark. More importantly, models with coefficients near unit (which leads to non-stationarity and non-invertibility) considerably decreased the performance of the bootstrap procedures. Though, MBB applied to models with lower parameter values seems to exhibit good properties.

In short, the Moving Block Bootstrap bias corrected estimate of confidence intervals seems to provide similar results than their bias corrected parameter estimate counterpart. This work can be improved by considering shorter block lengths where the Gaussian interval is expected to fail. However, an improvement might be achieved through the selection of an appropriate block length. Nevertheless, this is a cumbersome task, as the number of models to test is considerable and the choices for L increase with the length of the series. Thus, plug-in estimates or other formal devices for estimating the optimal value of L are of paramount importance.

Moreover, other resampling schemes might do a better job than the MBB. Therefore, research could focus not only on the other block resampling methods, but also on the DWB or MEB for instance. However, the empirical work must be followed by sound theoretical grounds and for this reason consistency properties of those different approaches must be established in the specific case of the GARMA model. In the case of confidence interval estimation, other procedures, besides the ones followed here, might me more suited.

Remarkably, the proposed **Algorithm 2** seems as a useful tool in selecting the optimal block length in the context of the MBB and multi-parameter estimation setting. The achieved decrease in computing time seems to outweigh the possible loss in estimating the **HHJ** algorithm in a restricted parameter space. This line of work is useful in areas such as Machine

Learning, where an automatized selection process is usually desired. In this line of work, the **NPPI** estimator of Lahiri, Furukawa and Lee (2007) can also be applied in nonparametric curve estimation problems, where there is a widespread usage of cross-validation.

The real data examples reinforce the utility of the MBB resampling scheme, however, when dealing with time series special care must be taken when estimating model parameters, as we might have to cope with non-stationarity and non-invertibility of the underlying stochastic processes. Moreover, as the second example shows, further research in the field of dependent bootstrap methods in relation to *GARMA* models have clear spill-over effects over other areas, for instance take the case of Finance/Econometrics with the *GARCH* family of models relation.

Notwithstanding, plug-in methods for block length selection might enhance the performance of the MBB confidence intervals and parameter bias estimation. In addition, other resampling schemes might also achieve this desired result, regarded that they are followed by theoretical grounds. Further, in terms of coverage rates, different bootstrap confidence interval procedures might show a clear improvement over the asymptotic one. In this line of work the sequential Monte Carlo method of Silva (2017) for interval estimation could be employed and intervals with guaranteed confidence coefficients stablished.

It is important to acknowledge that the conclusions of this work are limited to the class of *GARMA* models and more strictly to the low complexity models studied here. Owing to this, further research could focus on models of higher order shedding light into more general properties of the MBB in the context of the *GARMA* model. Besides, the effect of a varying linear term could also be studied, regarding the possible correlation between exogenous regressors and autoregressive or moving average terms.

This work can also be extended by an application of the procedures adopted here to the other members of the exponential family. Additionally, not only Bayesian estimation can be implemented, building on the work of Andrade (2016), but also a Bayesian Bootstrap (Rubin, 1981) framework. In the latter, a simulation of the posterior distribution of the parameter is computed, being operationally and inferentially similar to the traditional bootstrap.

REFERENCES

Andrade, B. S., Leslow, J. and Andrade, M. G. 2016. Transformed GARMA model: Properties and simulations. *Communications in Statistics - Simulation and Computation.* 2016.

Andrade, B. S., Andrade, M. G. and Ehlers, R. S. 2016. Bayesian Transformed GARMA Models. *Cornell University Library.* [Online] 2016. [Cited: 10 25, 2017.] https://arxiv.org/abs/1612.09561.

Andrade, B. S. 2016. GARMA models, a new perspective using Bayesian methods and transformations. *São Carlos : Estatística Interinstitucional do ICMC e UFSCarr.* [Online] Universidade de São Paulo, 2016. [Cited: 10 19, 2017.] http://www.teses.usp.br/teses/disponiveis/104/104131/tde-27032017-161141/.

Andrade, M. G., Ehlers, R. S. and Andrade, B. S. 2016. Bayesian GARMA Models for Count Data. *Communications in Statistics: Case Studies, Data Analysis and Applications.* 1, 2016, pp. 192-205.

Benjamin, M. A., Rigby, R. A. and Stasinopoulos, D. M. 2003. Generalized Autoregressive Moving Average Models. *Journal of the American Statistical Association.* Mar, 2003, Vol. 98, 461, pp. 214-223.

Bollerslev, T. 1986. GENERALIZED AUTOREGRESSIVE CONDITIONAL HETEROSKEDASTICITY. *Journal of Econometrics.* 31, 1986, pp. 307-327.

Box, G.E.P. and Jenkins, G.M. 1976. *Time Series Analysis: Forecasting and Control, Revised Edition.* San Francisco : Holden-Day, 1976.

Briet, O. J.T., Amerasinghe, P. H. and Vounatsou, P. 2013. Generalized Seasonal Autoregressive Integrated Moving Average Models for Count Data with Application to Malaria Time Series with Low Case Numbers. *PLOS ONE.* June, 2013, Vol. 8, 6.

Carlstein, E. 1986. THE USE OF SUBSERIES VALUES FOR ESTIMATING THE VARIANCE OF A GENERAL STATISTIC FROM A STATIONARY SEQUENCE. *The Annals of Statistics.* 1986, Vol. 14, 3, pp. 1171-1179.

Chernick, M. R. and LaBudde, R. A. 2011. *An introduction to bootstrap methods with applications to R.* New Jersey : John Wiley & Sons, Inc, 2011. ISBN 978-0-470-46704-6.

Chernick, M. R. 2008. *Bootstrap Methods: a guide for practitioners and researchers.* 2nd ed. New Jersey : John Wiley & Sons, 2008. ISBN 978-0-471-75621-7.

Cox, D. R., Gudmundsson, G., Lindgren, G., Bondesson, L., Harsaae, E., Laake, P., Juselius, K. and Lauritzen, S. L. 1981. Statistical Analysis of Time Series: Some Recent Developments [with Discussion and Reply]. *Scandinavian Journal of Statistics.* 1981, Vol. 8, 2, pp. 93-115.

Davis, R. A., Dunsmuir, T. M. and Wang, Y. 1999. Modelling time series of cunt data. [book auth.] S. Ghosh. *Asymptotics, Nomparametric & Time Series.* New York : Marcel Dekker, 1999, pp. 63-114.

DiCiccio, T. J. e Efron, B. 1996. Bootstrap Confidence Intervals. *Statistical Science.* 1996, Vol. 11, 3, pp. 189-228.

Durbin, J. and Koopman, S. J. 1197. Monte Carlo Maximum Likelihood Estimation for Non-Gaussian State Space Models. *Biometrika.* Sep., 1197, Vol. 84, 3, pp. 669-684.

—. 2000. Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives. *Journal of the Royal Statistical Society Series B.* 2000, Vol. 62, 1, pp. 3-56.

Efron, B. and Tibshirani, R. 1986. Bootstrap Methods for Standard Errors, Confidente Intervals, and Other Measures of Statistical Accuracy. *Statistical Science.* Feb., 1986, Vol. 1, 1, pp. 54-75.

Efron, B. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics.* Jan, 1979, Vol. 7, 1, pp. 1-26.

—. 1980. *THE JACKKNIFE, THE BOOTSTRAP, AND OTHER RESAMPLING PLANS.* STANFORD : TECHNICAL REPORT NO. 163, 1980.

—. 1983. Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. *Journal of the American Statistical Association.* Jun., 1983, Vol. 78, 382, pp. 316-334.

—. 1986. How Biased is the Apparent Error Rate of a Prediction Rule? *Journal of the American Statistical Association.* Jun., 1986, Vol. 81, 394, pp. 461-470.

Engle, R. F. 1982. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of UnitedKingdom Inflation. *Econometrica.* Jul., 1982, Vol. 50, 4, pp. 987-1007.

Engle, R. F., Lilien, D. M. e Robins, R. P. 1987. ESTIMATING TIME VARYING RISK PREMIA IN THE TERM STRUCTURE: THE ARCH-M MODEL. *Econometrica.* March, 1987, Vol. 55, 2.

Glosten, L. R., Jagannathan, R. e Runkle, D. E. 1993. On the Relation between the Expected Value and the Volatility of the Nominal ExcessReturn on Stocks. *The Journal of Finance,.* Dec., 1993, Vol. 48, 5.

Grunwald, G. K., Hyndma, R. J., Tedesco, L. and Tweedie, R. L. . 2000. NON-GAUSSIAN CONDITIONAL LINEAR AR(1) MODELS. *Australian & New Zealand Journal of Statistics.* December, 2000, Vol. 42, 4, pp. 479-495.

Hall, P., Horowitz, J. L e Jing, B-Y. 1995. On blocking rules for the bootstrap with dependent data. *Biometrika.* 82, 1995, pp. 561-574.

Hall, P. 1992. *The Bootstrap and Edgeworth Expansion.* New York : Springer-Verlag, 1992. ISBN 0-387-94508-3.

Jung, R. C., Kukuk, M, e Liesenfeld, R. 2006. Time series of count data: modeling, estimation and diagnostics. *Computational Statistics & Data Analysis.* 51, 2006, pp. 2350-2364.

Kedem, B. e Fokianos, K. 2002. *Regression Models for Time Series Analysis.* New Jersey : John Wiley & Sons, 2002. ISBN 0-471-36355-3.

Kunsch, Hans R. 1989. The Jakknife and the Bootstrap For General Stationary Observations. *The Annals of Statistics.* 1989, Vol. 17, 3, pp. 1217-1241.

Lahiri, S. N. 1999. THEORETICAL COMPARISONS OF BLOCK BOOTSTRAP METHODS. *The Annals of Statistics.* 1, 1999, Vol. 27, pp. 386-404.

—.2002. ON THE JACKKNIFE-AFTER-BOOTSTRAP METHOD FOR DEPENDENT DATA AND ITS CONSISTENCY PROPERTIES. *Econometric Theory.* 18, 2002, pp. 79-98.

—. 2003. *Resampling Methods for Dependent Data.* New York : Springer-Verlag, 2003.

Lahiri, S., Furukawa, K. and Lee, Yd. 2007. A nonparametric plug-in rule for selecting optimal block lengths for block bootstrap methods. *Statistical Methodology.* July, 2007, Vol. 4, 3, pp. 292-321.

Li, W. K. 1994. Time Series Models Based on Generalized Linear Models: Some Further Results. *Biometrics.* Jun., 1994, Vol. 50, 2, pp. 506-5011.

McCullagh, P. and Nelder, J. A. 1989. *Generalized Linear Models.* 2nd ed. s.l. : Chapman and Hall, 1989.

Peña, D., Tiao, G. C. and Tsay, R. S. 2001. *A course in time series analysis.* New York : John Wiley & Sons, 2001. ISBN 0-471-36164-X.

Rubin, D. B. 1981. The Bayesian Bootstrap. *The Annals of Statistics.* Vol. 9, 1, pp. 130-134

Shao, J. and Tu, D. 1995. *The Jakknife and Bootstrap.* New York : Springer-Verlag, 1995. ISBN 978-1-4612-6903-8.

Shao, X. 2010. The Dependent Wild Bootstrap. *Journal of the American Statistical Association.* March, 2010, Vol. 105, 489, pp. 218-235.

Silva, I. R. 2017. Confidece intervals through sequential Monte Carlo. *Computational Statistics and Data Analysis.* August, 2017, Vol. 105, 489, pp. 112-124.

Tsay, R. S. 2002. *Analysis of Financial Time Series.* s.l. : John Wiley & Sons, Inc, 2002. ISBN 0-471-41544-8.

Vinod, H. D. 2004. Ranking mutual funds using unconventional utility theory and stochastic dominance. *Journal of Empirical Finance.* 11, 2004, pp. 353-377.

Vinod, H. D. 2006. Maximum entropy ensembles for time series inference in economics. *Journal of Asian Economics.* 17, 2006, pp. 955-978.

Woodard, D. B., Matterson, D. S. e Henderson, S. G. 2011. Stationarity of generalized autoregressive moving average models. *Electronic Journal of Statistics.* 2011, Vol. 5, pp. 800-826.

Zeger, S. L. and Qaqish, B. 1988. Markov Regression Models for Time Series: A Quasi-Likelihood Approach. *BIOMETRICS.* December, 1988, Vol. 44, pp. 1019-1031.

Zheng, T., Xiao, H. and Chen, R. 2015. Generalized ARMA models with martingale difference errors. *Journal of Econometrics.* 2015, Vol. 189, 2, pp. 492-506.

APPENDIX I: TABLES

Table 5: Bootstrapped confidence intervals **coverage**, 1000 sim., 1000 boot. resamples, series of length 1000.

Model 1: Poisson-GARMA (1,0)

β=2; φ=0,15

length	norm	bias c.	perc.	basic
20	95.3%	94.5%	94.8% *	94.3%
	95.5%	92.4%	94.5%	92.6%
50	95.3%	94.1%	94.3%	93.8%
	95.5%	93.1%	93.7%	93.0%
100	95.3%	91.8%	91.8%	91.6%
	95.5%	91.4%	92.2%	91.3%

Model 2: Poisson-GARMA (1,0)

β=2; φ=0,5

length	norm	bias c.	perc.	basic
20	94.2%	92.3%	93.5%	92.1%
	93.8%	74.5%	74.8%	73%
50	94.2%	92.3%	92.7%	92.2%
	93.8%	88.1%	89.7%	87.5%
100	94.2%	90.7%	91%	90.9%
	93.8%	89.8%	90.5%	88.7%

Model 3: Poisson-GARMA (0,1)

β=2; θ=0,5

length	norm	bias c.	perc.	basic
20	93.7%	93.3%	93.9% *	93.2%
	79.6%	93.3% *	30.1%	92.8% *
50	93.7%	92.7%	93.3%	92.9%
	79.6%	85.7% *	72.0%	86.3% *
100	93.7%	91.2%	92.6%	91.7%
	79.6%	75.6%	79.1%	75.8%

Model 4: Poisson-GARMA (0,1)

β=2; θ=0,15

length	norm	bias c.	perc.	basic
20	95.5%	94.79% *	95.3% *	94.59% *
	94.6%	93.4%	93.3%	92.7%
50	95.5%	93.4%	94.4%	93.6%
	94.6%	92.8%	93.5%	92.0%
100	95.5%	92.0%	92.0%	91.6%
	94.6%	90.9%	92.1%	90.2%

Model 5: Poisson-GARMA (1,1)

β=2; φ=0,5; θ=0,1

length	norm	bias c.	perc.	basic
20	94.4%	93.3%	93.5%	92.6%
	94.0%	78.7%	90.7%	76.2%
	95.5%	92.5%	98.4%	92.5%
50	94.4%	92.6%	93.2%	92.6%
	94.0%	87.7%	94.7% *	86.1%
	95.5%	92.3%	96.8%	92.2%
100	94.4%	91.8%	91.9%	91.2%
	94.0%	88.5%	93.9%	86.6%
	95.5%	91.6%	95.7%	91.1%

Model 6: Gamma-GARMA (1,0)

β=2; φ=0,15; σ=1,41

length	norm	bias c.	perc.	basic
20	94.1%	93.9%	93.2%	93.4%
	95.1%	92.6%	94.4%	92.5%
	95.2%	94.2%	94.7%	94.3%
50	94.1%	93.4%	92.8%	93.1%
	95.1%	93.6%	93.8%	93.4%
	95.2%	93.3%	94.0%	93.0%
100	94.1%	91.7%	91.6%	91.2%
	95.1%	91.9%	91.8%	91.7%
	95.2%	91.6%	91.5%	91.9%

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal (i.e. norm.wbc)

Table 6: Bootstrapped confidence intervals **coverage**, 1000 sim., 1000 boot. resamples, series of length 1000.

Model 7: Gamma-GARMA (1,0)

β=2; φ=0,5; σ=1,41

length	norm	bias c.	perc.	basic
	94.1%	90.8%	89%	91.3%
20	95.1%	82.8%	75.1%	75.8%
	95.1%	87.6%	89.8%	86.8%
	94.1%	92.9%	91.3%	92.8%
50	95.1%	93.0%	90.1%	91.3%
	95.1%	93.0%	93.8%	92.6%
	94.1%	91.7%	90.7%	91.6%
100	95.1%	94.2%	90.8%	92.6%
	95.1%	92.4%	92.3%	92.0%

Model 8: Gamma-GARMA (0,1)

β=2; θ=0,5; σ=1,41

length	norm	bias c.	perc.	basic
	93.4%	93.5% *	93.8% *	93.9% *
20	94.4%	87.1%	61.5%	58.0%
	94.3%	87.0%	87.2%	84.7%
	93.4%	94.3% *	93.6% *	93.8% *
50	94.4%	97.6%	91.1%	86.2%
	94.3%	93.4%	93.3%	92.4%
	93.4%	91.6%	91.5%	91.1%
100	94.4%	98.4%	93.9%	90.4%
	94.3%	93.3%	92.7%	91.1%

Model 9: Gamma-GARMA (0,1)

β=2; θ=0,15; σ=1,41

length	norm	bias c.	perc.	basic
	91.9%	92.9% *	93.8% *	91.8%
20	93.2%	95.4% *	94.5% *	91.2%
	95.0%	94.4%	94.5%	94.0%
	91.9%	92.6% *	92.5% *	92% *
50	93.2%	96.1% *	95.1% *	93.5% *
	95.0%	93.0%	93.9%	92.5%
	91.9%	91.0%	91.6%	89.5%
100	93.2%	95.4% *	94.1% *	92.3%
	95.0%	92.5%	91.4%	91.9%

Model 10: Gamma-GARMA (1,1)

β=2; φ=0,5; θ=0,1; σ=1,41

length	norm	bias c.	perc.	basic
	93.8%	90.4%	93.5%	90.6%
20	92.9%	94.2% *	98.4%	90.9%
	92.1%	96.6% *	99.9%	95.4% *
	95.2%	83.8%	85.4%	81.9%
	93.8%	91.0%	93.2%	90.6%
50	92.9%	94.1% *	98.4%	92.3%
	92.1%	94.2% *	99.1%	93.3% *
	95.2%	92.3%	93.5%	91.6%
	93.8%	90.1%	92.4%	90.0%
100	92.9%	92.3%	97.4%	90.4%
	92.1%	92.7% *	97.7% *	90.6%
	95.2%	92.5%	92.9%	91.8%

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal (i.e. norm.wbc)

Table 7: Bootstrapped confidence intervals **coverage**, 1000 sim., 1000 boot. resamples, series of length 30.

Model 1: Poisson-GARMA (1,0)

β=2; φ=0,15

length	norm	bias c.	perc.	basic
4	92.3%	86.1%	88.6%	85.7%
	95.6%	82%	93%	81.3%
7	92.3%	84%	86%	84.5%
	95.6%	82.2%	89.5%	79.8%
10	92.3%	80.8%	82%	79.9%
	95.6%	80.9%	86.2%	77.8%

Model 2: Poisson-GARMA (1,0)

β=2; φ=0,5

length	norm	bias c.	perc.	basic
4	90.7%	80.9%	83.9%	80.8%
	94.5%	76.4%	65.6%	72.4%
7	90.7%	80.7%	82.8%	80%
	94.5%	82.4%	69.1%	75.1%
10	90.7%	77.2%	79.4%	76.9%
	94.5%	81.5%	66.8%	74%

Model 3: Poisson-GARMA (0,1)

β=2; θ=0,5

length	norm	bias c.	perc.	basic
4	85.1%	83.9%	89.1% *	85.3% *
	11.4%	48.5% *	6%	54.5% *
7	85.1%	81.6%	86% *	82.9%
	11.4%	40.9% *	12.4% *	47.7% *
10	85.1%	76.9%	82.4%	77%
	11.4%	36.6% *	14.6% *	42.1% *

Model 4: Poisson-GARMA (0,1)

β=2; θ=0,15

length	norm	bias c.	perc.	basic
4	90.9%	87.1%	89.2%	87.4%
	89.2%	81.7%	97.2% *	83.1%
7	90.9%	84.9%	86%	85%
	89.2%	80.9%	95.4% *	82.8%
10	90.9%	80%	81.9%	80.3%
	89.2%	80.1%	93.6% *	81%

Model 5: Poisson-GARMA (1,1)

β=2; φ=0,5; θ=0,1

length	norm	bias c.	perc.	basic
4	90.4%	79.3%	85.1%	78.4%
	95.5%	72.4%	97.6%	69.5%
	93.7%	86.5%	100%	86.1%
7	90.4%	79.8%	84.1%	78.4%
	95.5%	75.4%	95.9%	69.3%
	93.7%	84.9%	100%	85.4%
10	90.4%	76.3%	80.8%	75.1%
	95.5%	74.2%	95% *	67.4%
	93.7%	84.3%	100%	84.5%

Model 6: Gamma-GARMA (1,0)

β=2; φ=0,15; σ=1,41

length	norm	bias c.	perc.	basic
4	90.4%	84.4%	85.6%	84.3%
	94.1%	83.4%	94.8% *	80.4%
	90.9%	87.3%	84%	88.2%
7	90.4%	83.3%	84.3%	82.5%
	94.1%	83.4%	92.1%	79.9%
	90.9%	85.1%	80.3%	85%
10	90.4%	79.9%	80.7%	77.4%
	94.1%	82.1%	89.5%	77.5%
	90.9%	81.2%	77.4%	80.9%

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal (i.e. norm.wbc)

Table 8: Bootstrapped confidence intervals **coverage**, 1000 sim., 1000 boot. resamples, series of length 30.

Model 7: Gamma-GARMA (1,0)

β=2; φ=0,5; σ=1,41

length	norm	bias c.	perc.	basic
	84.3%	76.7%	75.1%	73.5%
4	92.4%	85.7%	78.6%	81.5%
	90.1%	83.5%	90.3% *	84.1%
	84.3%	76.2%	76.1%	75.5%
7	92.4%	87%	79.4%	82.2%
	90.1%	83.1%	87.6%	84%
	84.3%	72.3%	74.2%	71.1%
10	92.4%	85.2%	76.8%	78.7%
	90.1%	81%	84.8%	80.9%

Model 8: Gamma-GARMA (0,1)

β=2; θ=0,5; σ=1,41

length	norm	bias c.	perc.	basic
	90.1%	86.3%	83.1%	86.2%
4	89.9%	84.9%	96% *	82.5%
	91.3%	85.7%	90.2%	86%
	90.1%	84.8%	82.8%	84.3%
7	89.9%	89.4%	95.7% *	87.2%
	91.3%	84.8%	86.8%	84.9%
	90.1%	82.4%	80.1%	82.2%
10	89.9%	90.4% *	93.4% *	88.3%
	91.3%	82.1%	83.7%	81.5%

Model 9: Gamma-GARMA (0,1)

β=2; θ=0,15; σ=1,41

length	norm	bias c.	perc.	basic
	90.1%	86.9%	85.4%	84.3%
4	91.3%	88%	97.4% *	87.8%
	90.9%	88.4%	84.2%	88.5%
	90.1%	85.9%	83.7%	83.9%
7	91.3%	89.5%	94.9% *	88.3%
	90.9%	86%	79.8%	85.2%
	90.1%	83%	81.3%	80.2%
10	91.3%	89%	93.1% *	85.8%
	90.9%	82.3%	77.7%	81.1%

Model 10: Gamma-GARMA (1,1)

β=2; φ=0,5; θ=0,1; σ=1,41

length	norm	bias c.	perc.	basic
	82.6%	98.2% *	85.3% *	82%
4	88.4%	88.5% *	100% *	82.3%
	84%	87.7% *	100% *	81.7%
	88.6%	80.3%	88.4%	81.1%
	82.6%	95.4% *	82.5%	81%
7	88.4%	87.1%	100% *	80.3%
	84%	86.9% *	100% *	80.4%
	88.6%	80.2%	85.1%	80.6%
	82.6%	88.7% *	77.8%	76.8%
10	88.4%	85.2%	98.9% *	77%
	84%	86.8% *	100% *	79.5%
	88.6%	78.2%	82.8%	78.2%

Legend: fields marked with * are used to denote a coverage rate higher (in absolute difference to the 95% target) than the respective asymptotic normal (i.e. norm.wbc)

APPENDIX II: FIGURES

Figure 19: Model 1 parameters distribution, series of length 1000

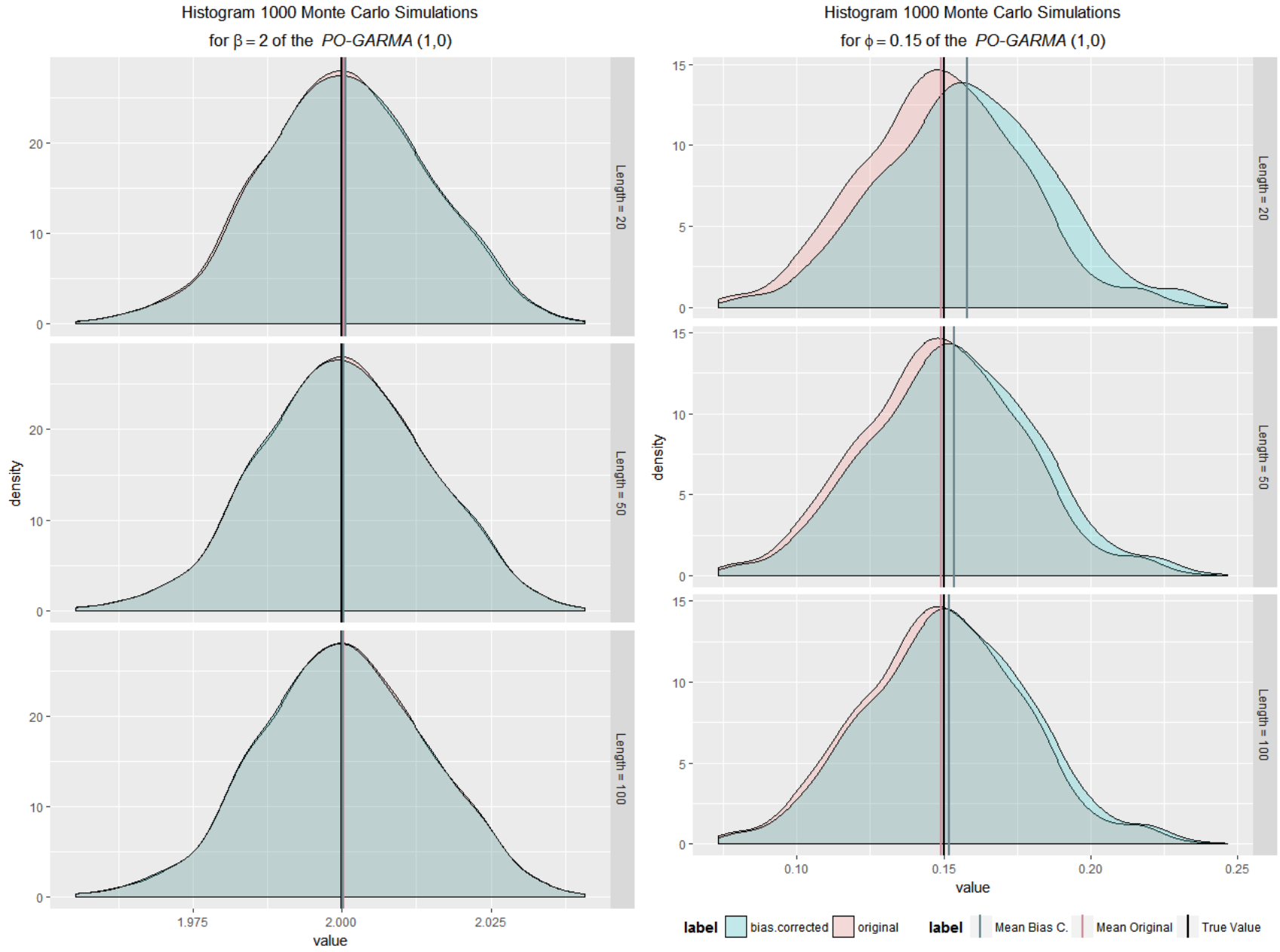


Figure 20: Model 2 parameters distribution, series of length 1000

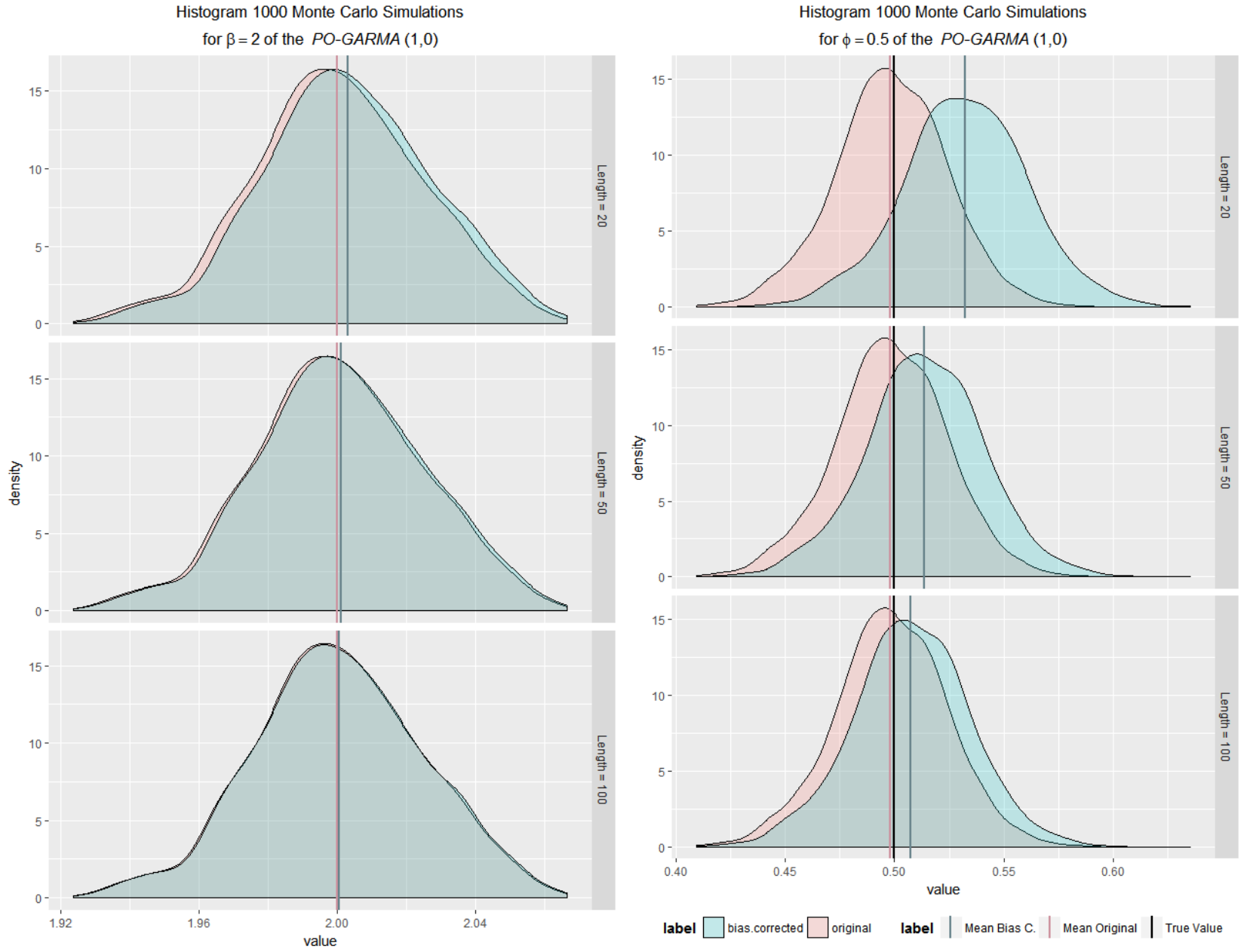


Figure 21: Model 3 parameters distribution, series of length 1000

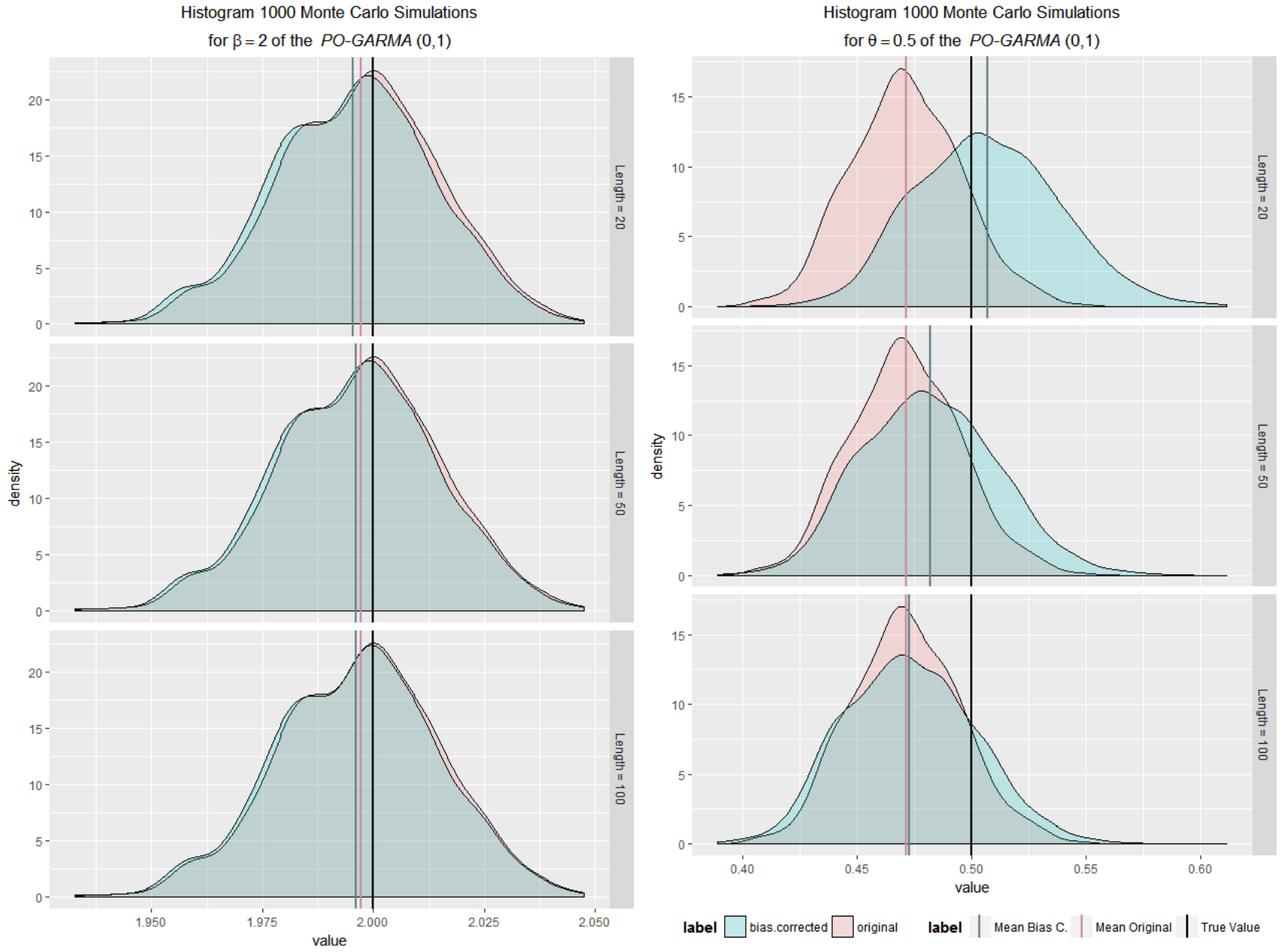


Figure 22: Model 4 parameters distribution, series of length 1000

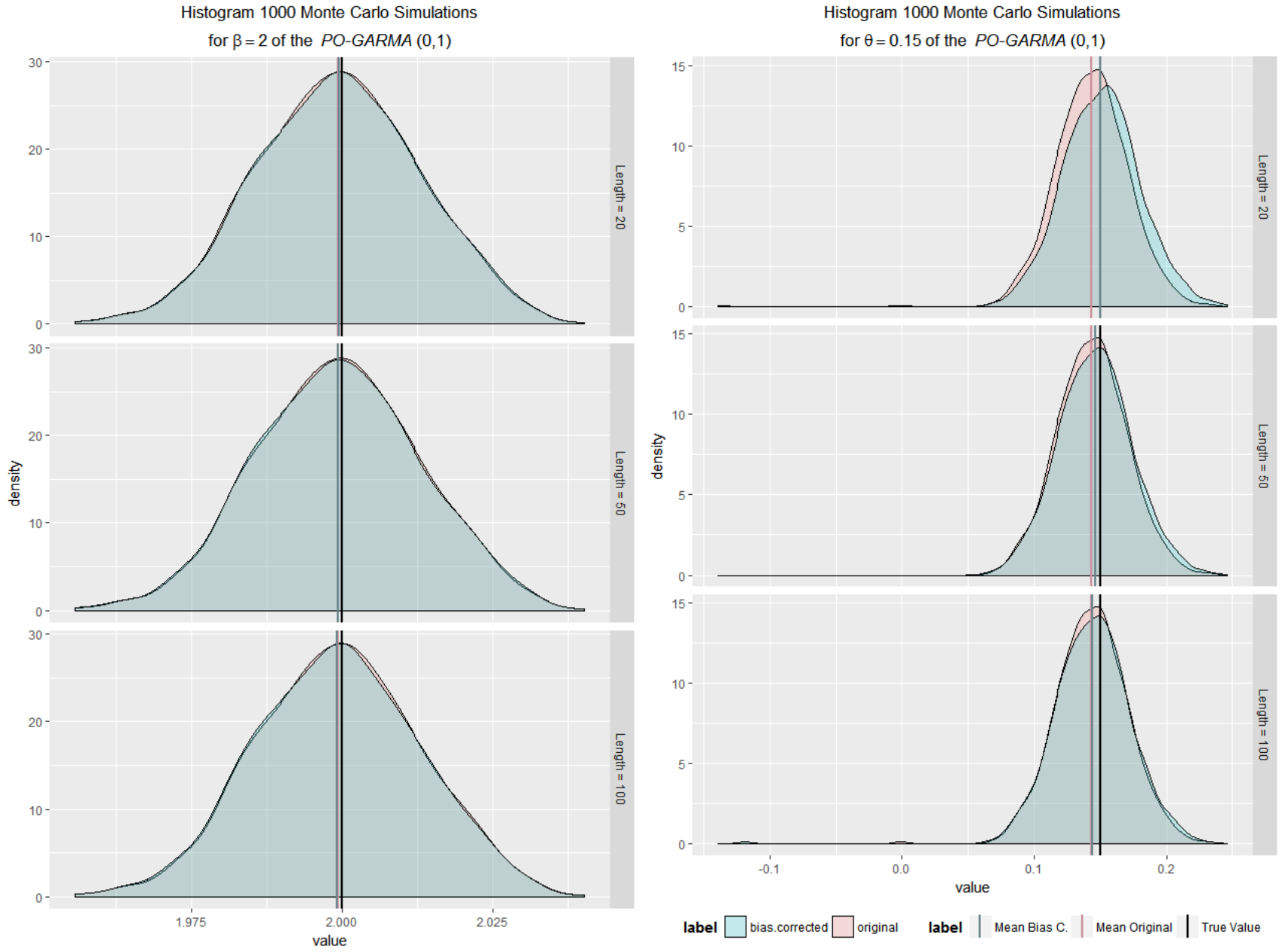


Figure 23: Model 5 parameters distribution, series of length 1000

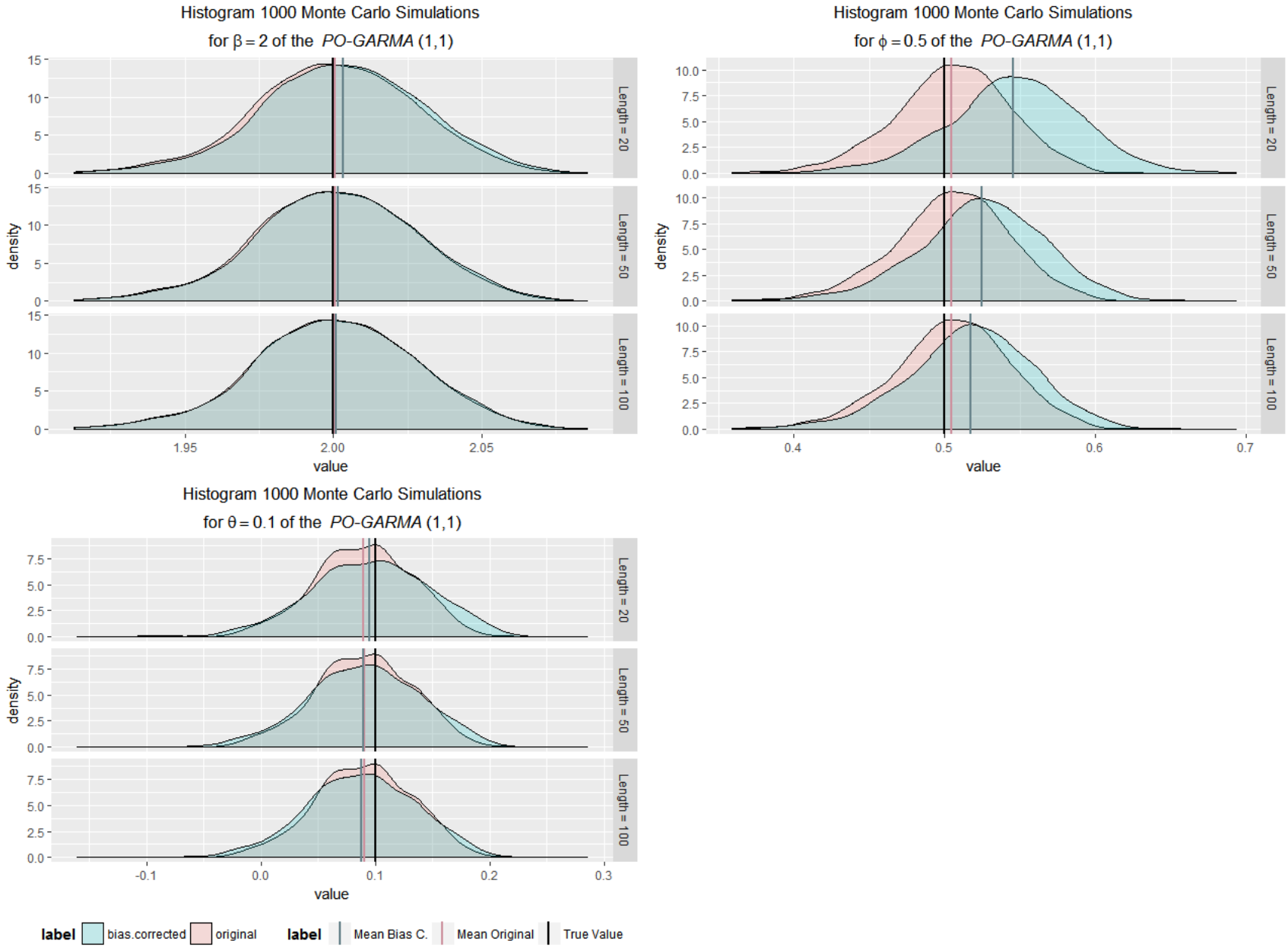


Figure 24: Model 6 parameters distribution, series of length 1000

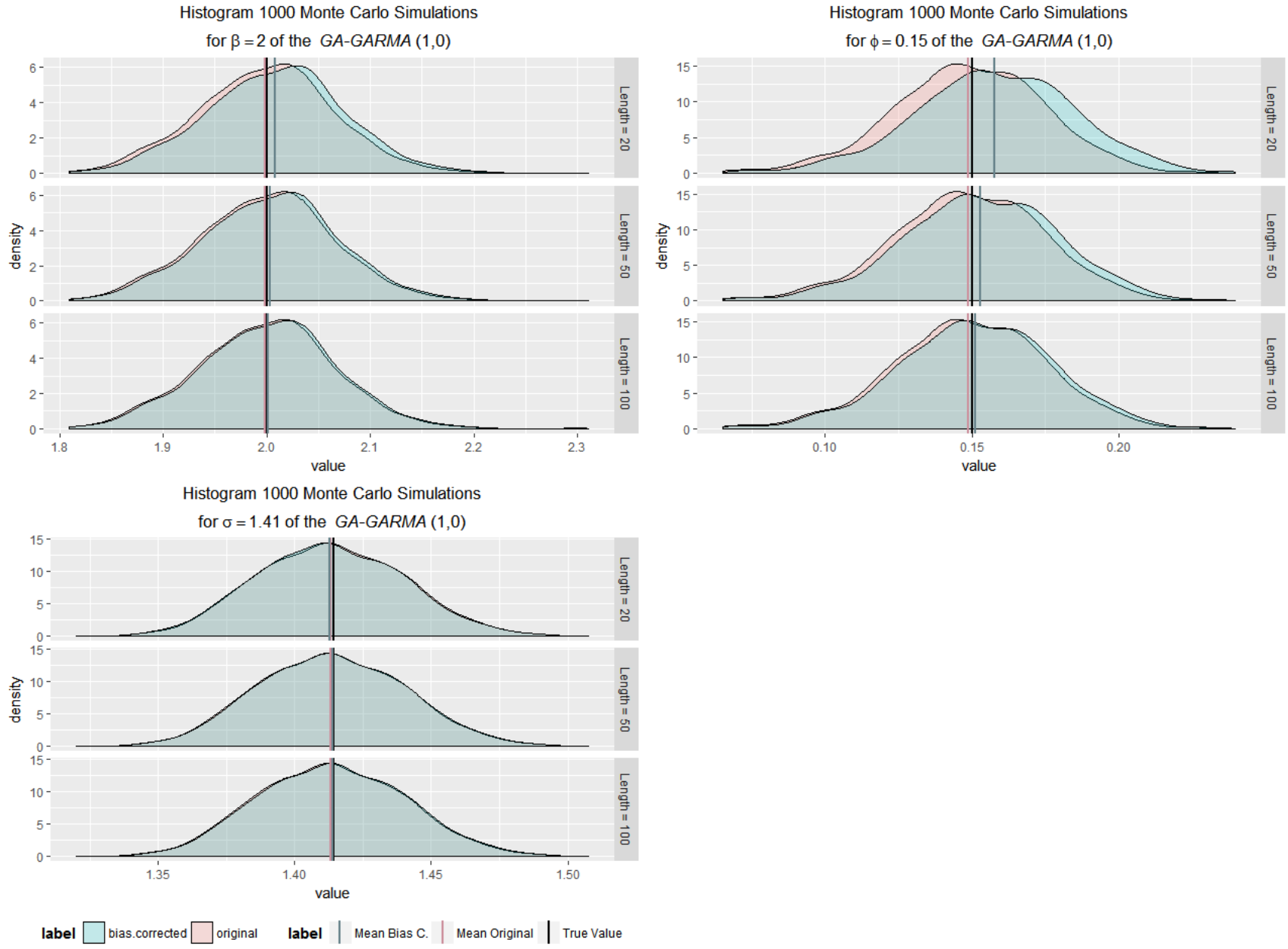


Figure 25: Model 7 parameters distribution, series of length 1000

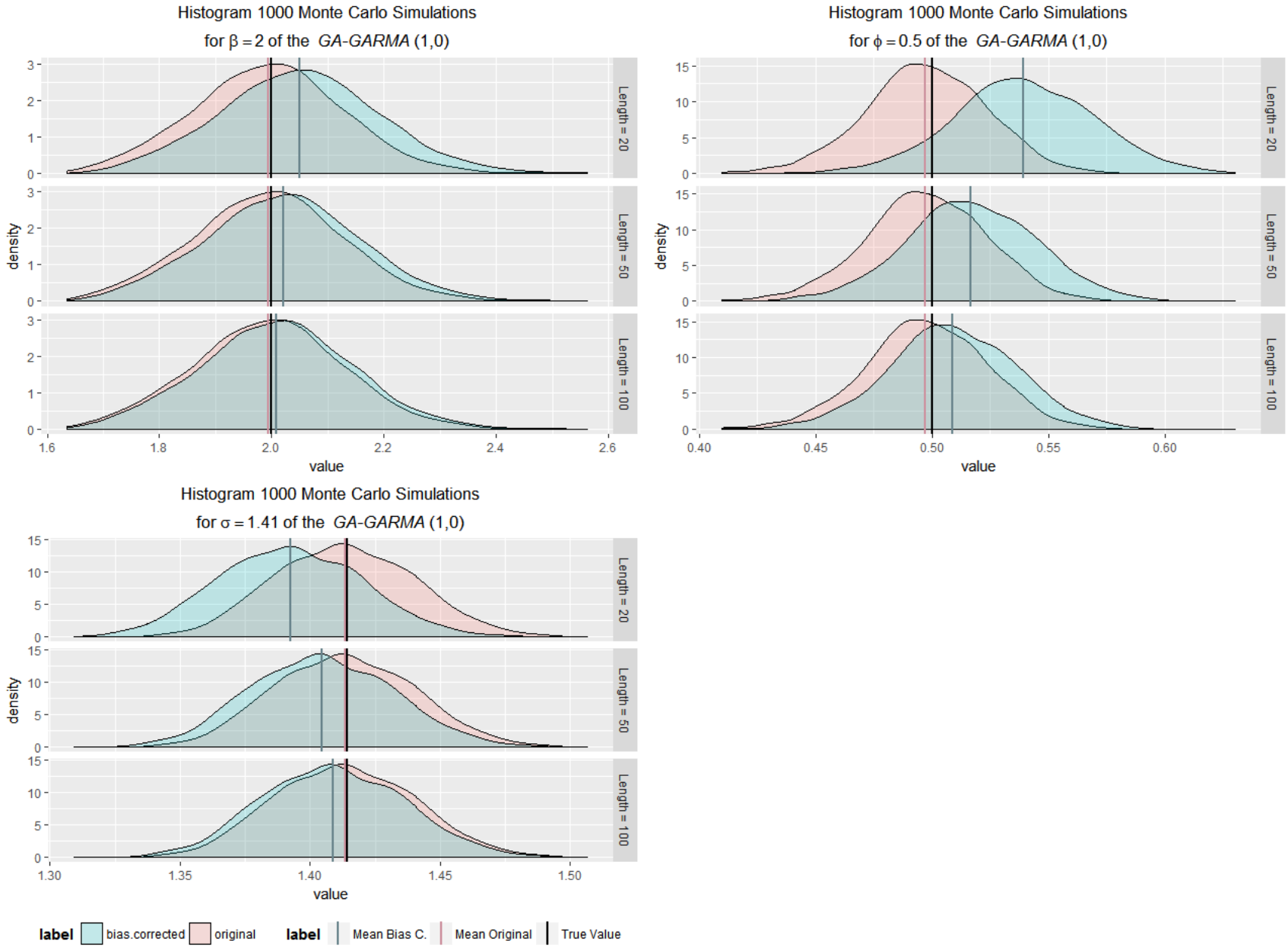


Figure 26: Model 8 parameters distribution, series of length 1000

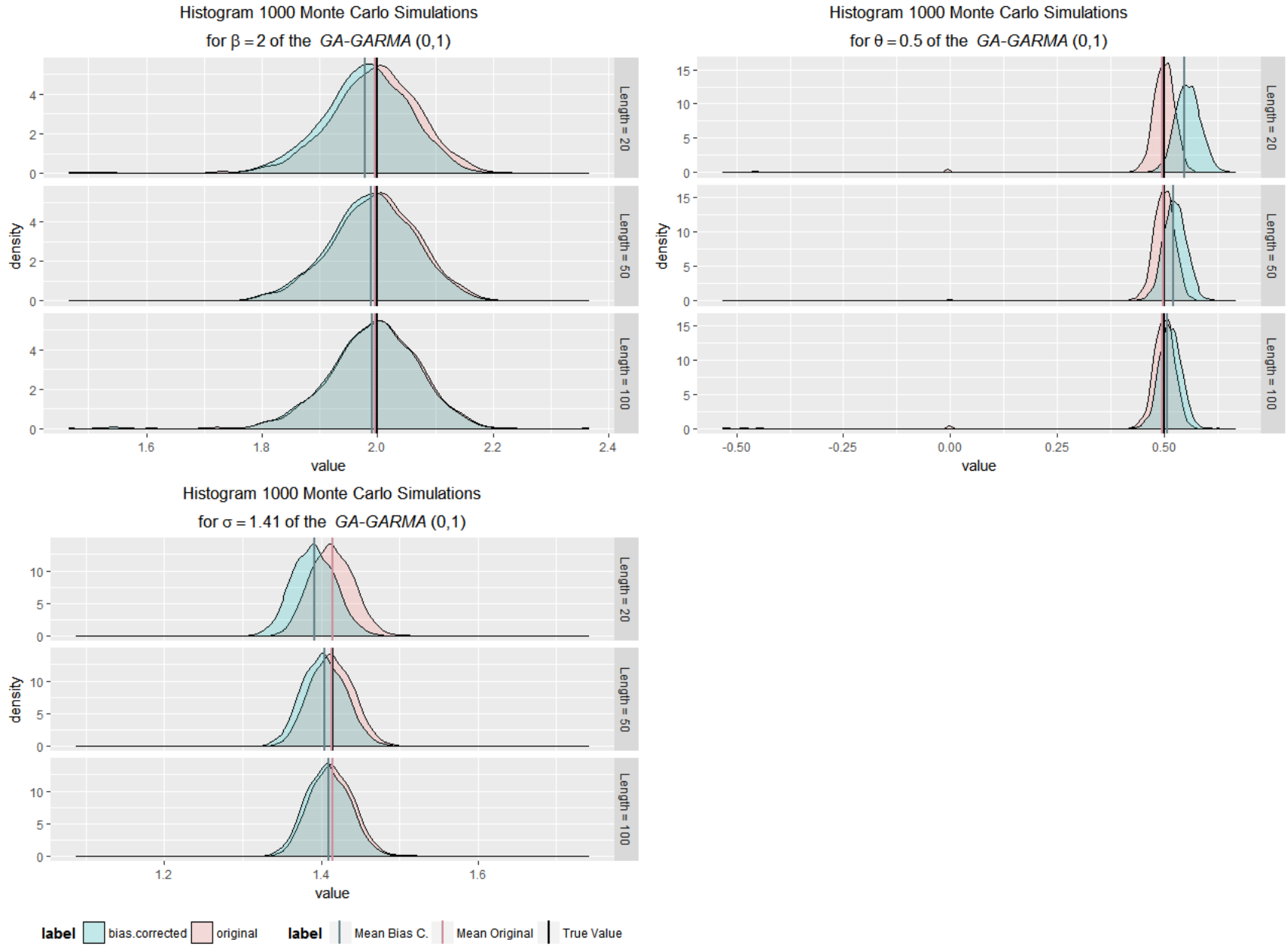


Figure 27: Model 9 parameters distribution, series of length 1000

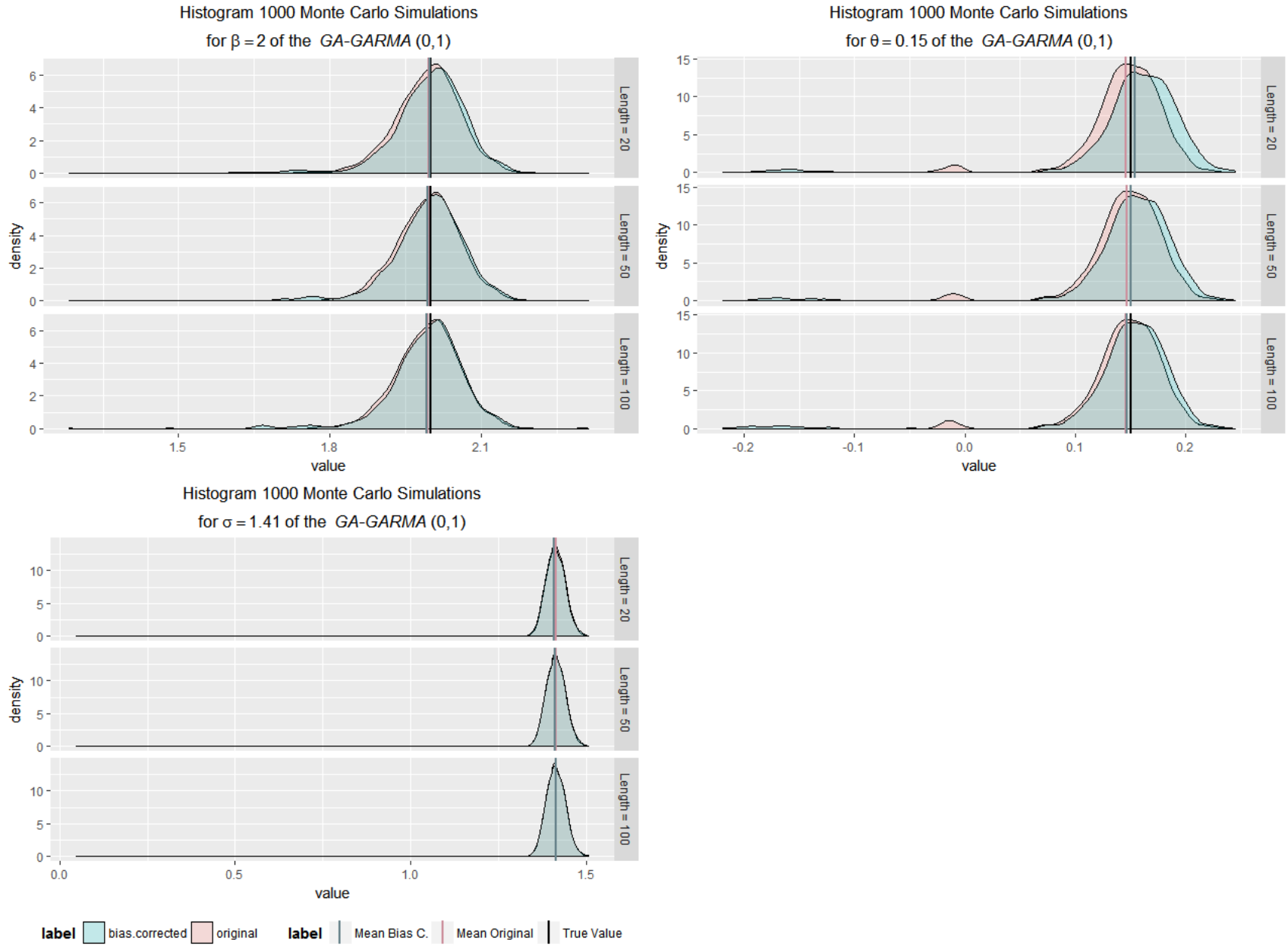


Figure 28: Model 10 parameters distribution, series of length 1000

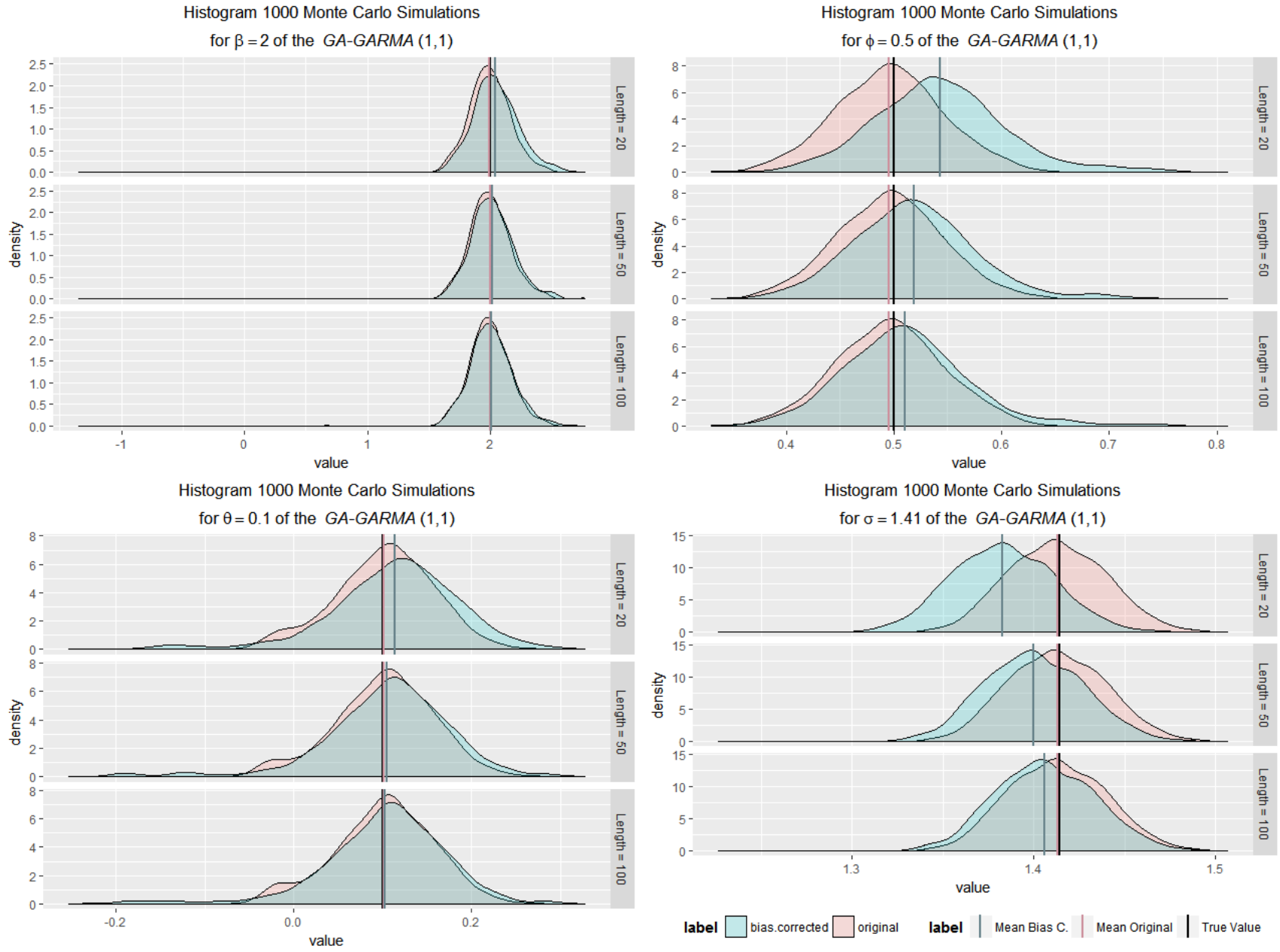


Figure 29: Model 1 parameters distribution, series of length 30

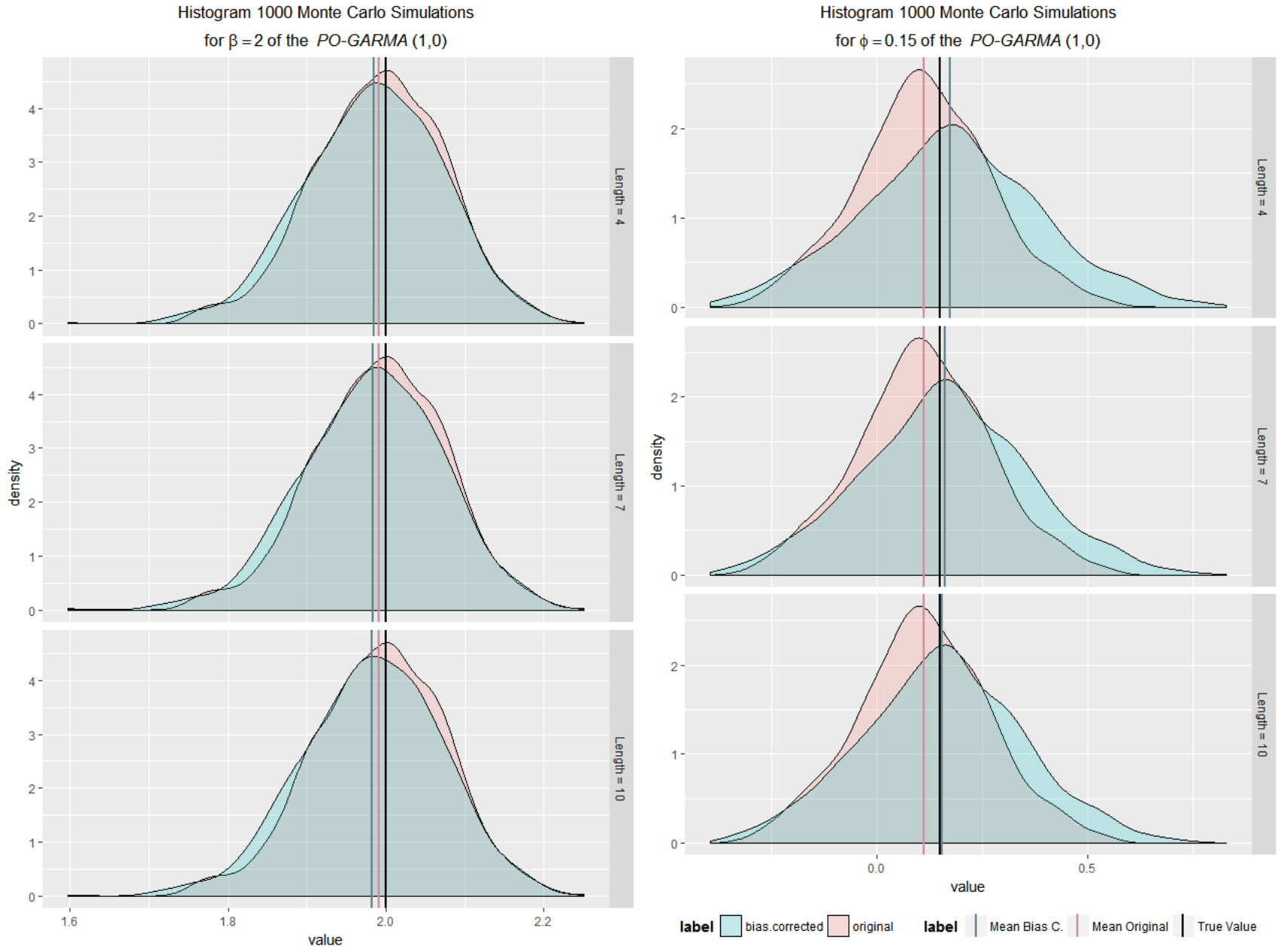


Figure 30: Model 2 parameters distribution, series of length 30

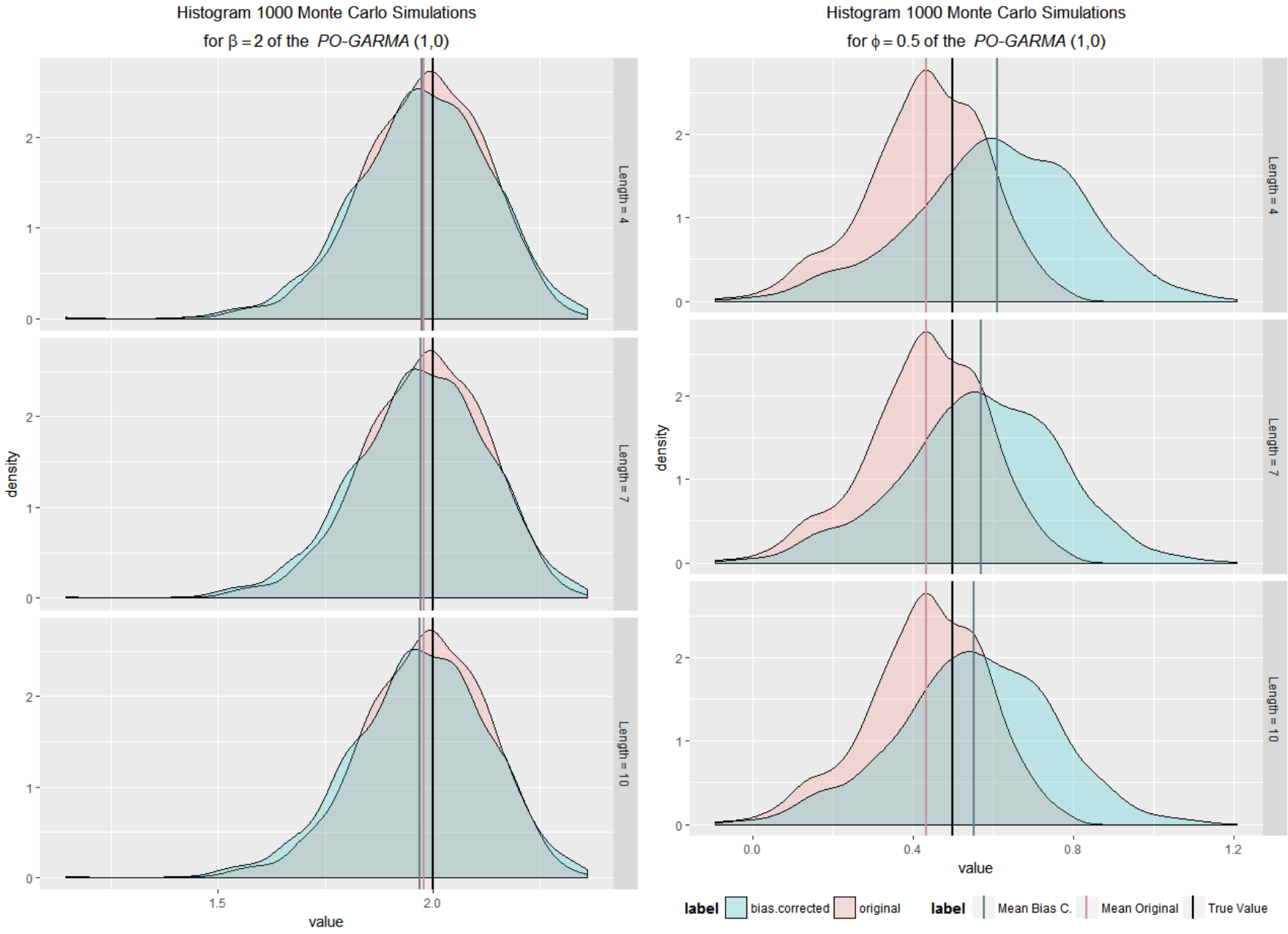


Figure 31: Model 3 parameters distribution, series of length 30

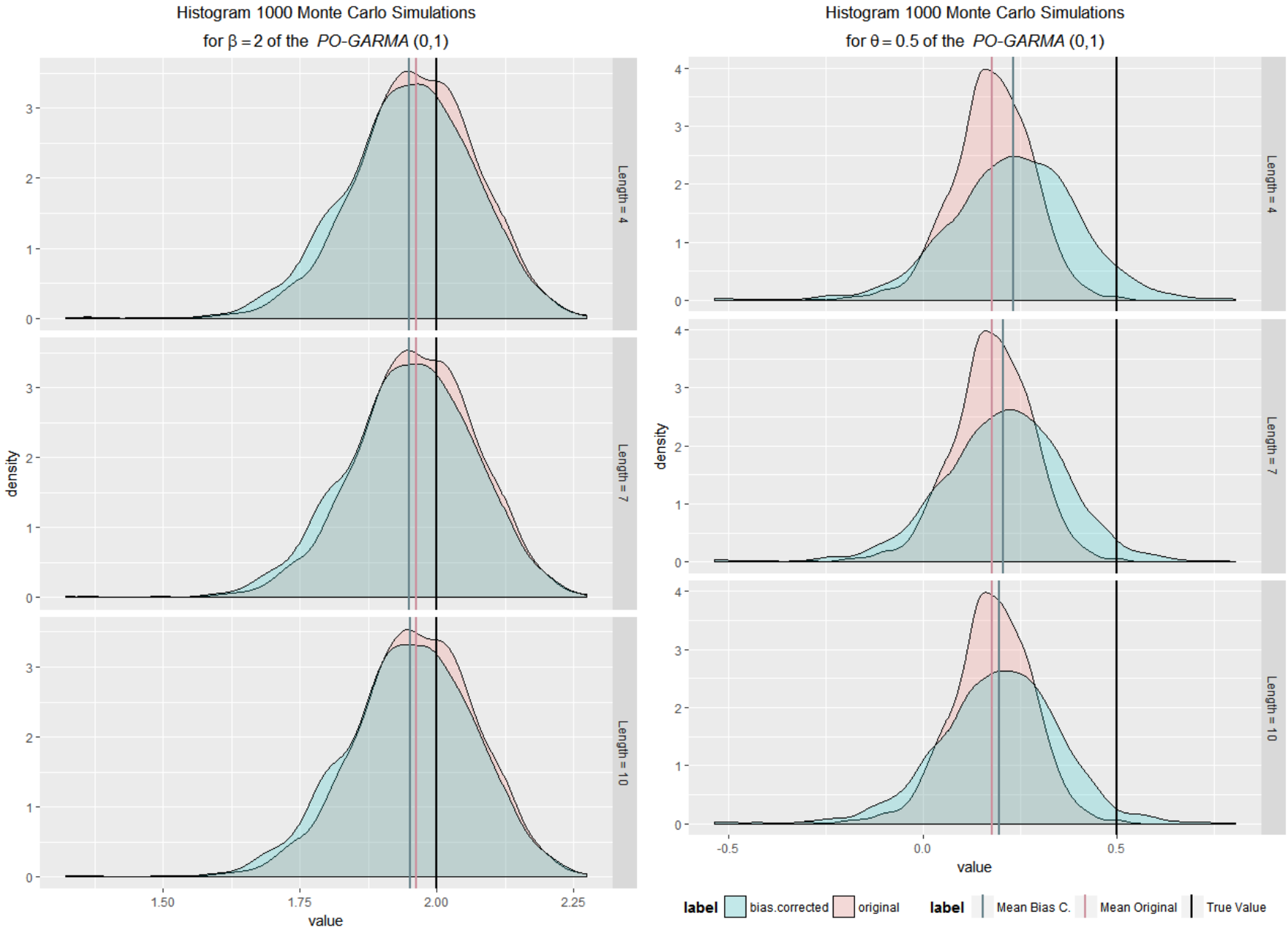


Figure 32: Model 4 parameters distribution, series of length 30

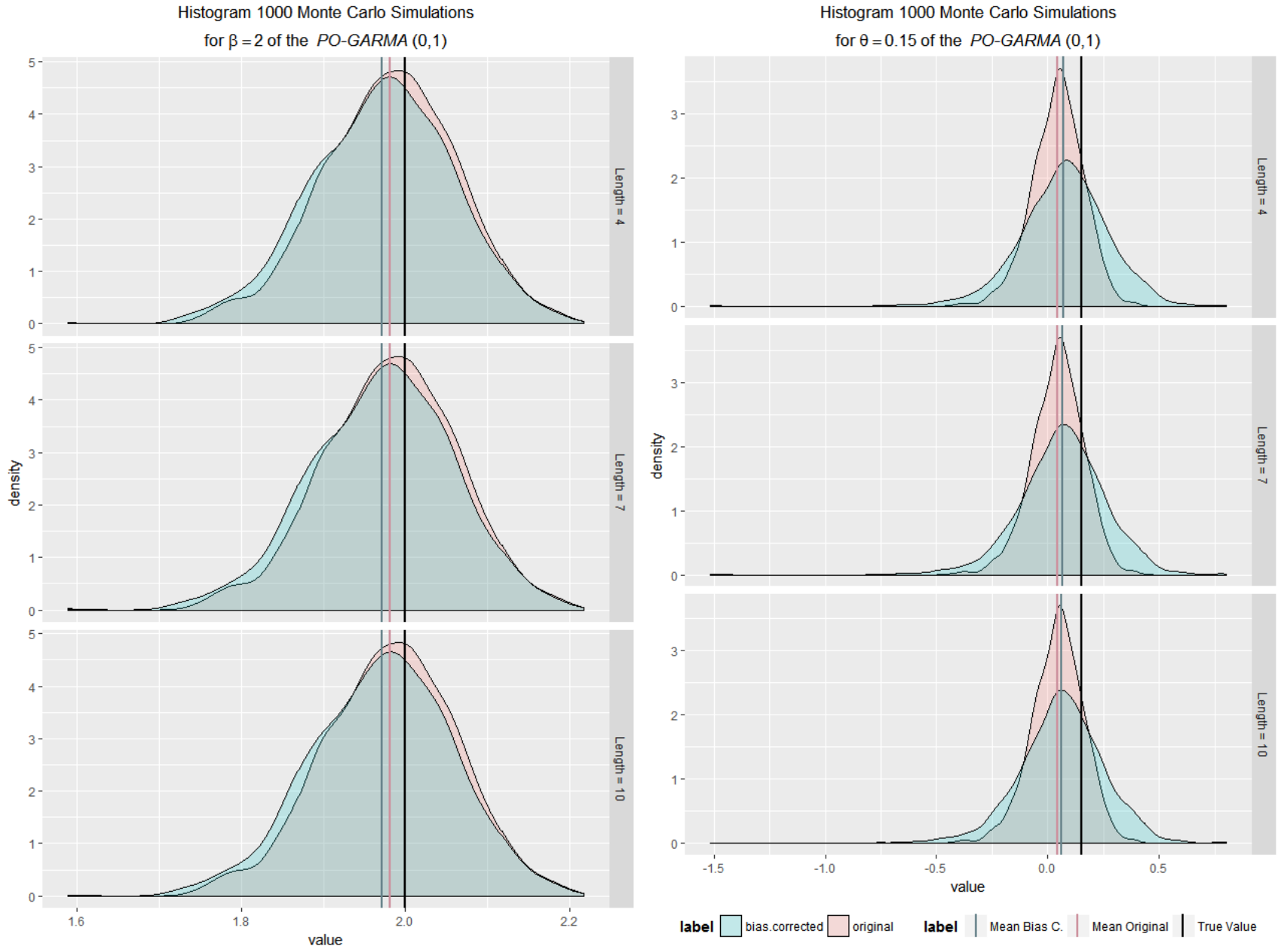


Figure 33: Model 5 parameters distribution, series of length 30

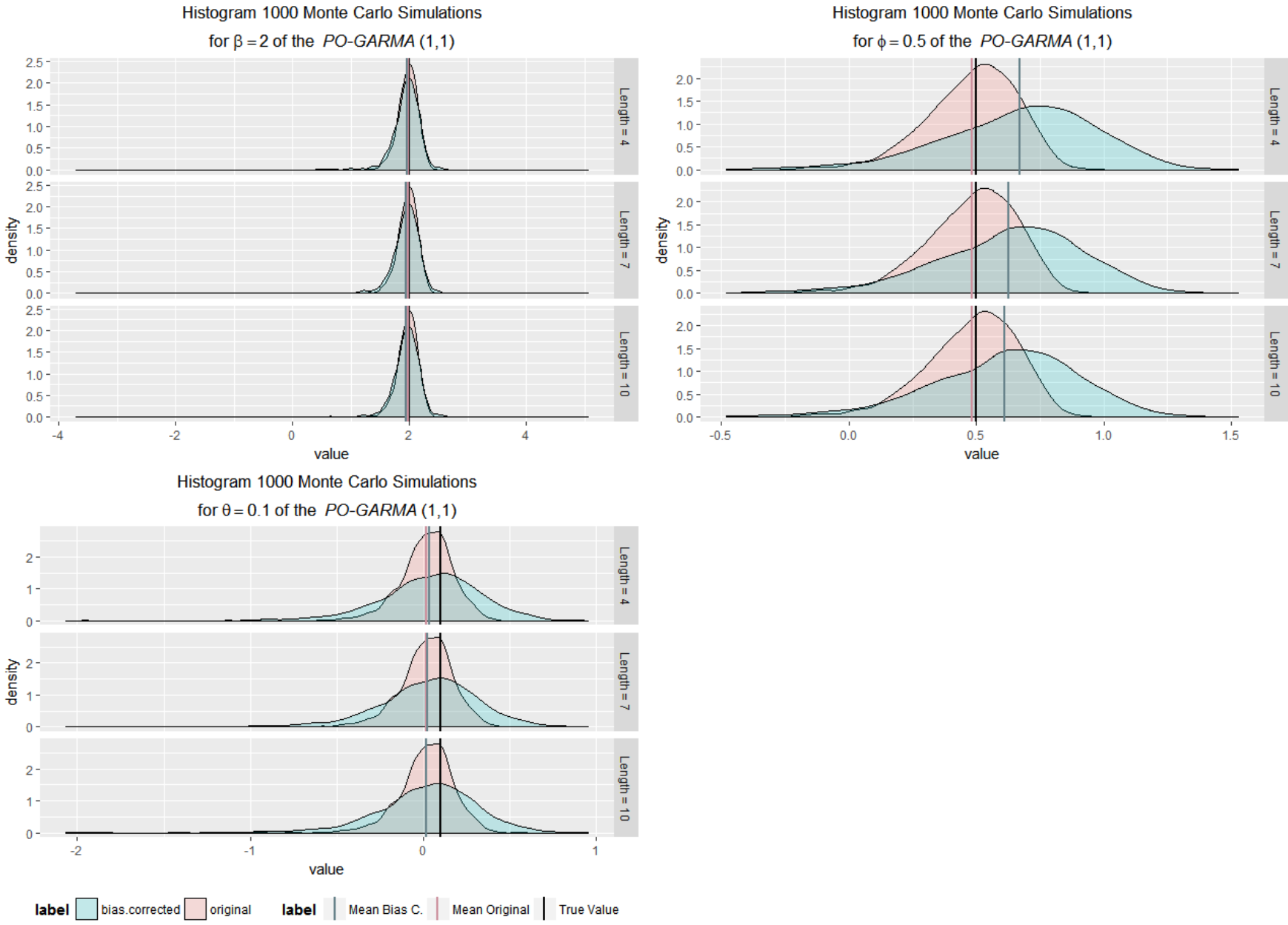


Figure 34: Model 6 parameters distribution, series of length 30

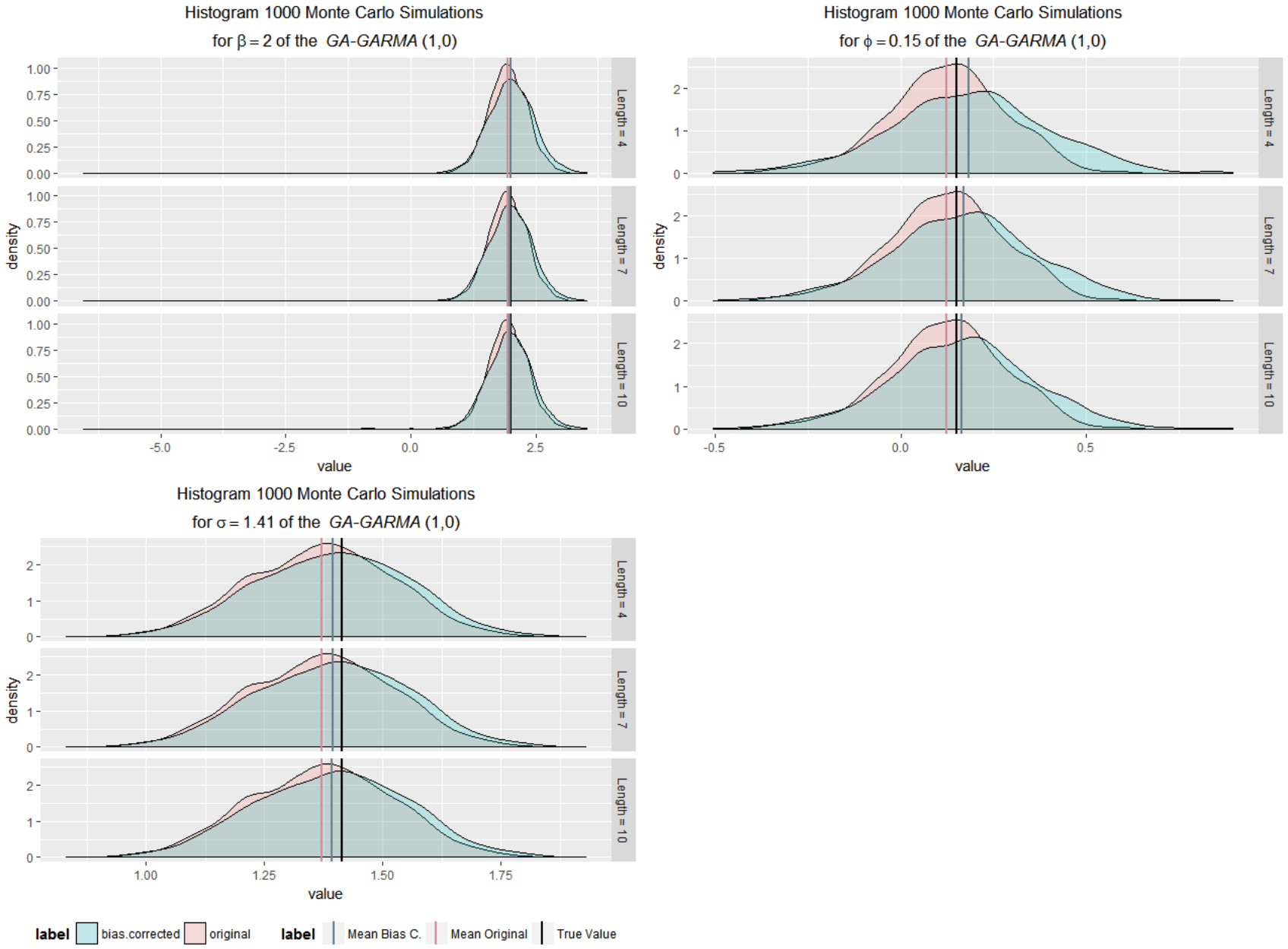


Figure 35: Model 7 parameters distribution, series of length 30

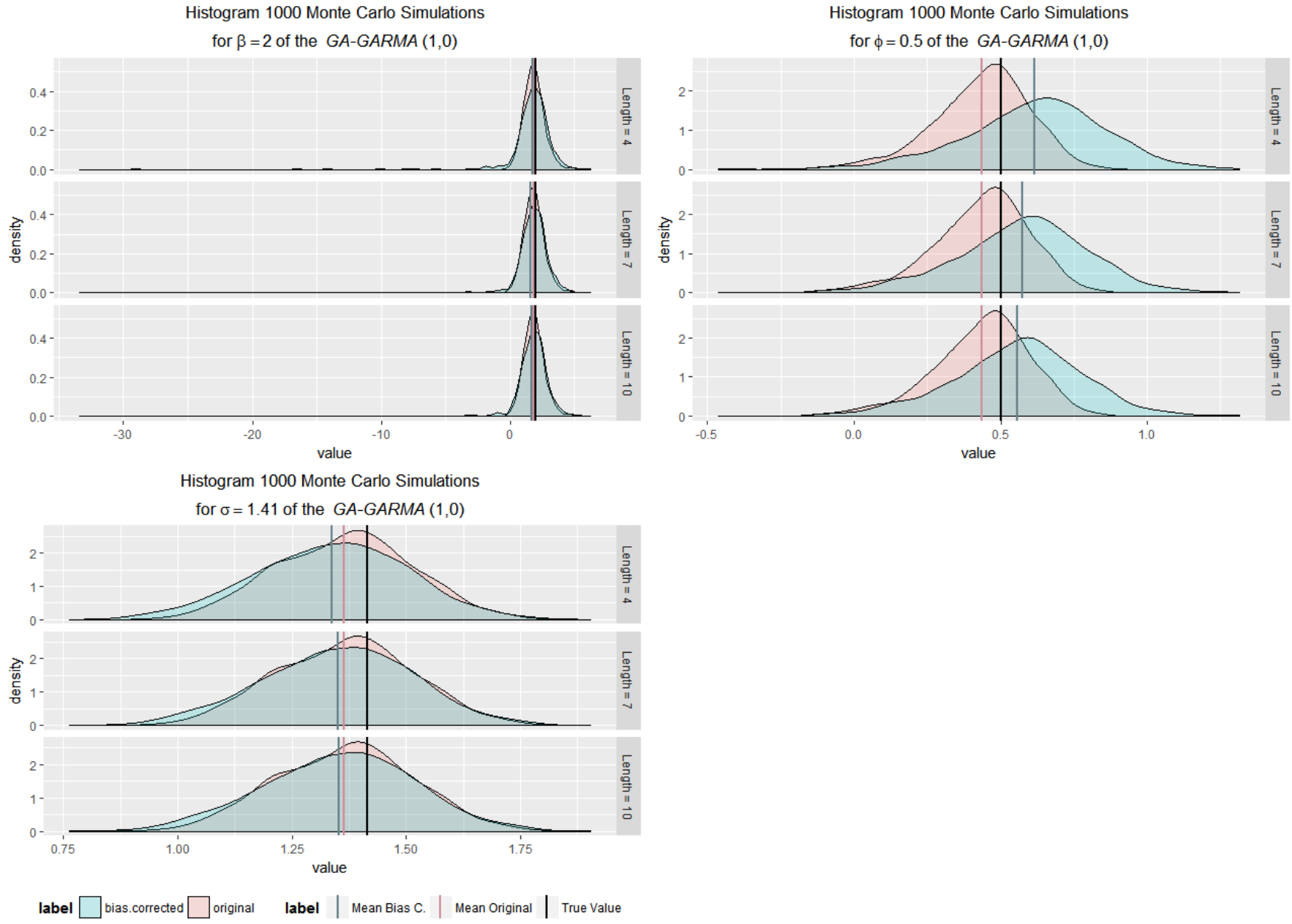


Figure 36: Model 8 parameters distribution, series of length 30

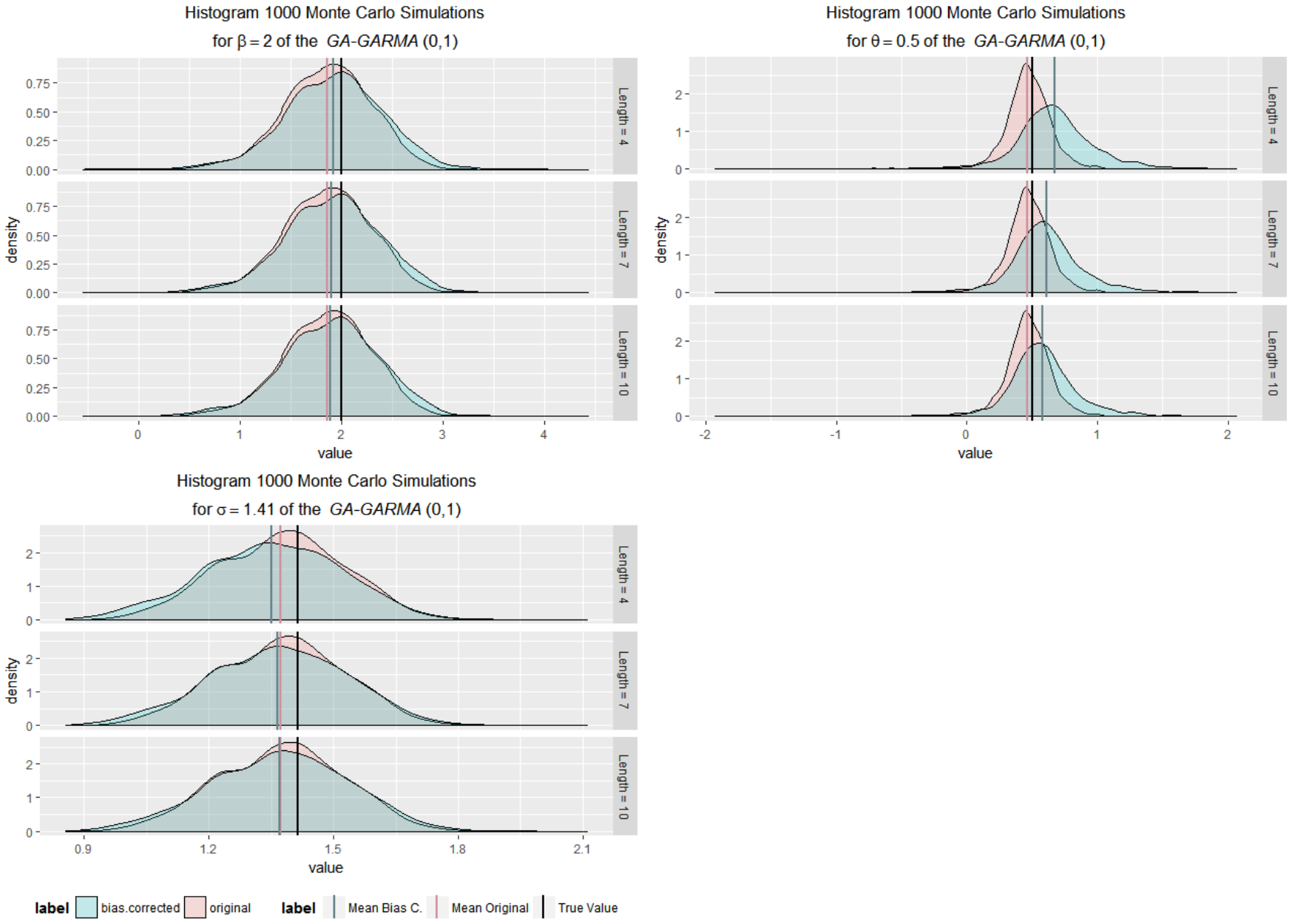


Figure 37: Model 9 parameters distribution, series of length 30

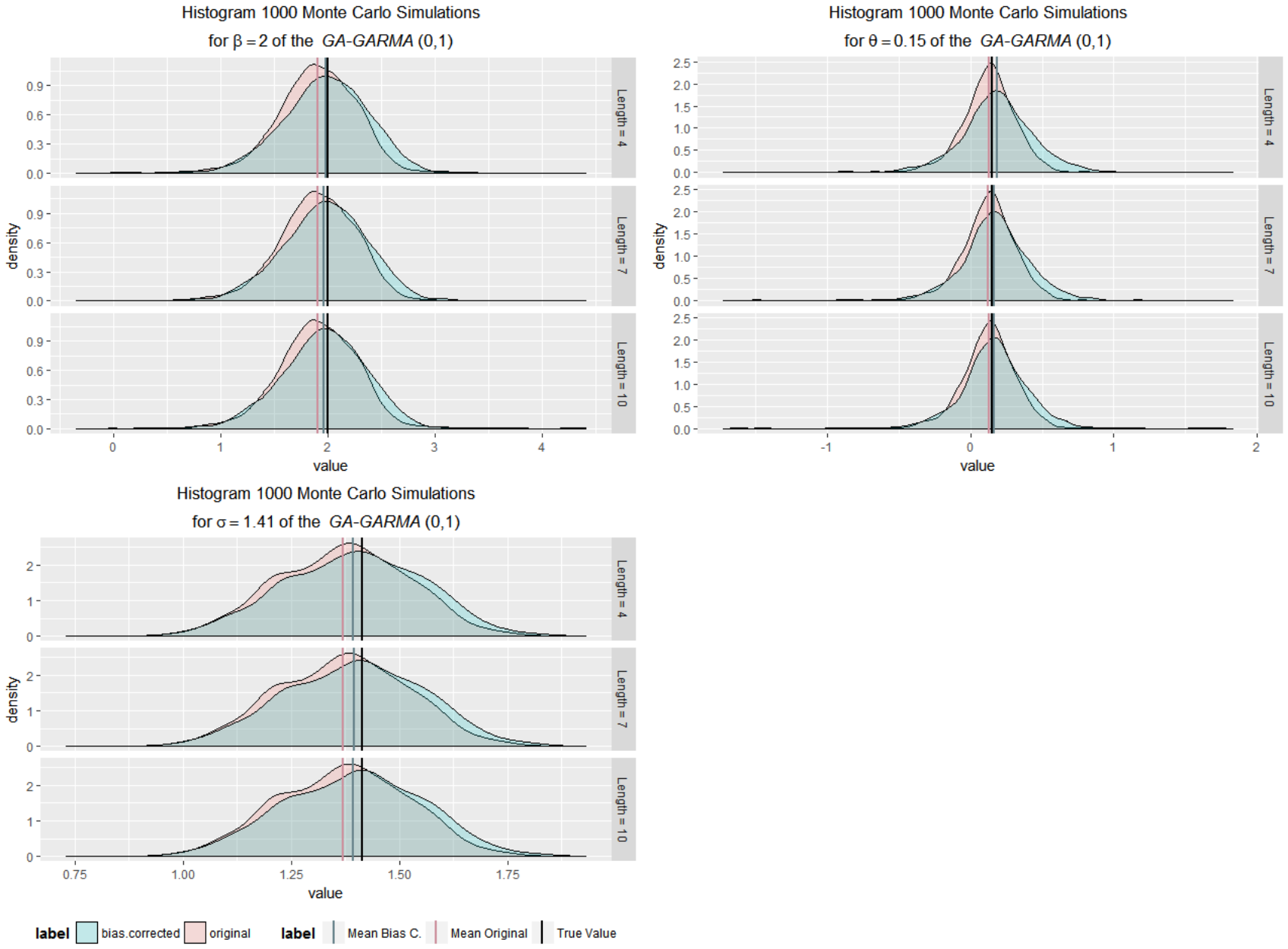
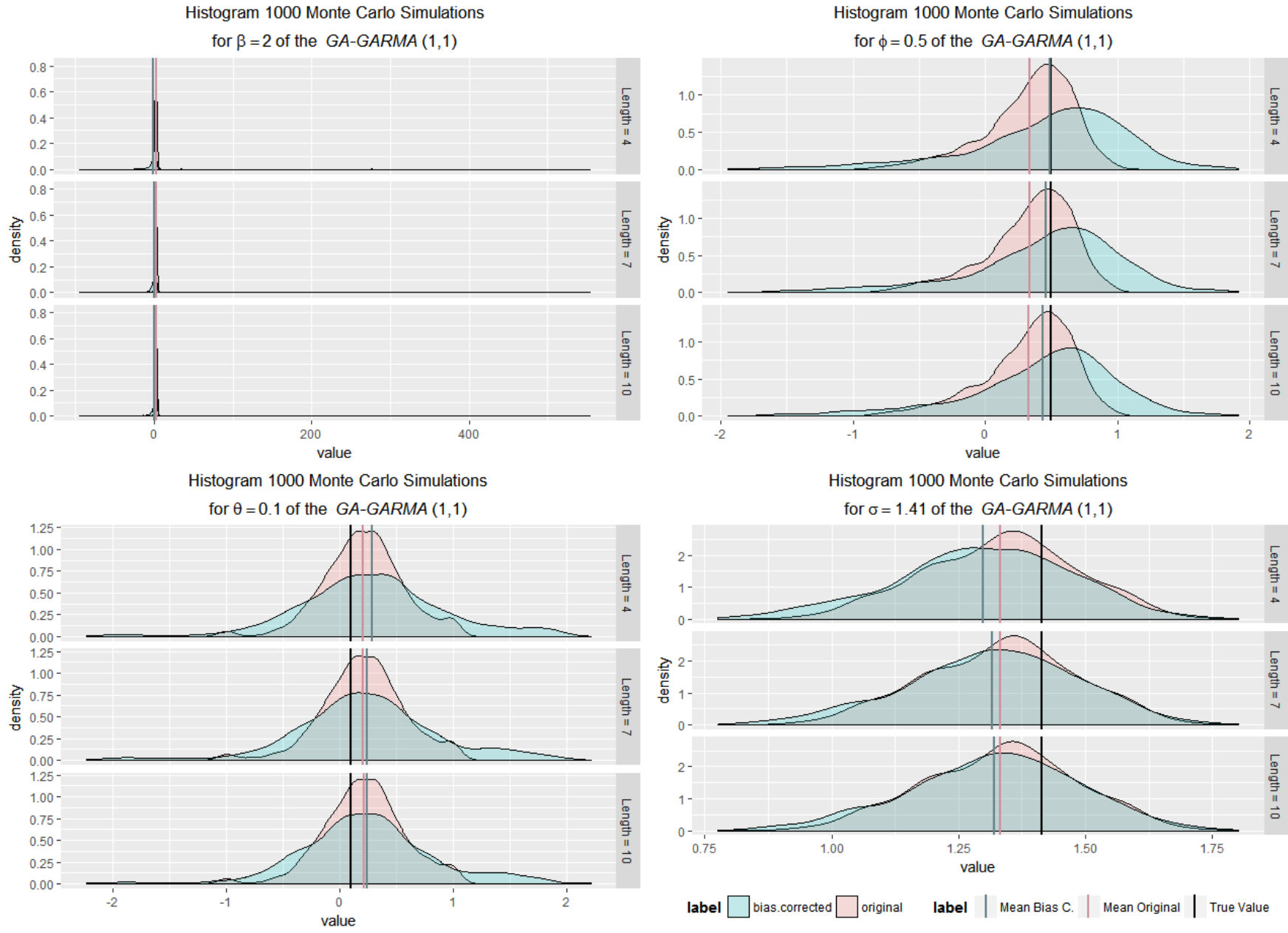


Figure 38: Model 10 parameters distribution, series of length 30



APPENDIX III: R CODE

R Code Master Thesis

Matheus de Vasconcellos Barroso

March 12, 2018

INTRODUCTION

The original code for the Master thesis was not really tidy, with almots 4000 lines of code in the main file. Also, with Monte Carlo simulations and MBB the number of failures in the estimation process was leaving tracktability difficult. To organize the ideas this R markdown file was generated in order to mimic all the steps required to recreate the thesis. Additionaly, two package were created: garma and dboot, that will be used througout this file.

The first step is to load both packages, their installation and help files can be found at https://github.com/matheusbarroso/garma and https://github.com/matheusbarroso/dboot.

1. Install:

```
install.packages('devtools')
library(devtools)
install_github("matheusbarroso/garma@v.1.0.2")
install_github("matheusbarroso/dboot@v.1.0.1")
```

2. Load:

```
library(dboot)
library(garma)
```

The second step is to come up with a list of models to use:

model	phi	theta	beta.x	mu0	sigma2	sigma	alpha	familia
1	0.15	0.00	2	10	NA	NA	0.1	PO
2	0.50	0.00	2	10	NA	NA	0.1	PO
3	0.00	0.50	2	10	NA	NA	0.1	PO
4	0.00	0.15	2	10	NA	NA	0.1	PO
5	0.50	0.10	2	10	NA	NA	0.1	PO
6	0.15	0.00	2	10	2	1.414214	0.1	GA
7	0.50	0.00	2	10	2	1.414214	0.1	GA
8	0.00	0.50	2	10	2	1.414214	0.1	GA
9	0.00	0.15	2	10	2	1.414214	0.1	GA
10	0.20	0.10	2	10	2	1.414214	0.1	GA

ANALYSIS SERIES OF LENGTH 1000

The last table was assigned to the object models (a data.frame), so that we can easily specify them as (note that `nrow(X) = 1001`, so we will be working with a series of length 1000):

```
specs <- lapply(seq_len(nrow(models)), function (j)
  switch(family = models$familia[j],
         "PO"={
           garma::GarmaSpec(family = "PO",
                     beta.x = 2,
                     phi = models$phi[j],
                     theta = models$theta[j],
                     mu0 = models$mu0[j],
                     X = as.matrix(data.frame(intercept = rep(1,1001)))
           )

         },
         "GA"={
           garma::GarmaSpec(family = "GA",
                     beta.x = 2,
                     phi = models$phi[j],
                     theta = models$theta[j],
                     mu0 = models$mu0[j],
                     sigma2 = models$sigma2[j] ,
                     X = as.matrix(data.frame(intercept = rep(1,1001)))
           )
         })

)
```

Generating 1000 Monte Carlo simulatins for each spec, series of length 1000:

```
#register the parallel back-end:

no_cores <- if(detectCores() == 1) 1 else detectCores() -1
registerDoParallel(no_cores)

sims <- lapply(specs, function (j)
  GarmaSim(spec = j,
             nmonte = 1000,
             nsteps = 1000,
             burnin = 0
             )
    )
```

The estimation is straightforward and also the Coefficients Distribution plots:

```
fits <- lapply(sims, garma::GarmaFit)

#for the first model:
plot(fits[[1]])
summary(fits[[1]])

#for the fitfh model:
```

```
plot(fits[[5]])
summary(fits[[5]])

#for the fitfh model:
plot(fits[[10]])
summary(fits[[10]])

#the final plot:
fig.4 <- GarmaFit(
  GarmaSim(
    GarmaSpec(
      family = "PO",
      beta.x = 2,
      phi = 0.8,
      mu0 = 10,
      X = as.matrix(data.frame(intercept = rep(1,1001)))
      ),
    nmonte = 1000,
    nsteps = 1000,
    burnin = 0
    )
  )


plot(fig.4)
summary(fig.4)
```

Moving on to the MBB replicates: (Read the warning message bellow)

```
boot.out <- lapply(sims,
                   GarmaSimBoot,
                   l = c(20,50,100),
                   R = 1000)

model <- 1

plot(boot.out[[model]], type = 'original-bias')
summary(boot.out[[model]])
```

Warning: please note that this task will *take too long to execute*, you should try a smaller example (`nmonte < 100` and/or `nsteps < 100`and/or `R < 100`. Also, the easiest way to go is to simply estimate one model). Here, we provide a 'toy' example:

```
no_cores <- if(detectCores() == 1) 1 else detectCores() -1
registerDoParallel(no_cores)

toy <- GarmaSimBoot(
  GarmaSim(
    GarmaSpec(
      family = "PO",
      beta.x = 2,
      phi = 0.2,
      mu0 = 10,
      X = as.matrix(data.frame(intercept = rep(1,1001)))
      ),
    nmonte = 10,
    nsteps = 1000,
```

```
    burnin = 0
    ),
  l = c(20,50,100),
  R = 30)

plot(toy, type='original-bias')
summary(toy)
```

The last step for the series of length 1000 is to estimate the coverage rates:

```
ci <- ConfidenceInterval(toy)
coverage(ci)
```

ANALYSIS SERIES OF LENGTH 30

Now to generete the series of length 30:

```
specs <- lapply(seq_len(nrow(models)), function (j)
  switch(family = models$familia[j],
         "PO"={
           garma::GarmaSpec(family = "PO",
                     beta.x = 2,
                     phi = models$phi[j],
                     theta = models$theta[j],
                     mu0 = models$mu0[j],
                     X = as.matrix(data.frame(intercept = rep(1,31)))
           )

         },
         "GA"={
           garma::GarmaSpec(family = "GA",
                     beta.x = 2,
                     phi = models$phi[j],
                     theta = models$theta[j],
                     mu0 = models$mu0[j],
                     sigma2 = models$sigma2[j] ,
                     X = as.matrix(data.frame(intercept = rep(1,31)))
           )
         })

)

sims <- lapply(specs, function (j)
  GarmaSim(spec = j,
             nmonte = 1,
             nsteps = 30,
             burnin = 0
             )
    )
```

Once more we are faced with the task of generating the MBB resamples / plots, *be aware* that the following code should take **long** to execute:

```
boot.out <- lapply(sims,
                   GarmaSimBoot,
                   l = c(4,7,10),
```

```
                    R = 1000)

model <- 1 # change here to produce the information for a #different model

plot(boot.out[[model]], type='original-bias')
summary(boot.out[[model]])
```

Now, to estimate the coverage rates:

```
ci <- lapply(boot.out, ConfidenceInterval)
coverage(ci[[model]]) #model 1...
```

Finally to generate all the MBB resamples from 1 to 20 (as the GarmaSimBoot already is working on parallel use simple outer nested loops):

```
resamp <- lapply(seq_len(20),
                 function(j)
                   lapply(sims, GarmaSimBoot, l = j))
```

The plots (note that you should change the *model* flag with values from 1 to 10 to the individuals plot, you can also loop for all plots):

```
library(plyr)
temp <- lapply(seq_len(10),
               function(i) {
                 ldply(seq_len(20),
                       .fun = function(j) {
                         resamp[[j]][[i]]@print.out
                         }
                 )
                           }
               )
for (i in seq_len(10)) names(temp[[i]])[3] <- "value"

model <- 2 #switch from 1 to 10 to a different plot or loop #through them...

temp[[model]]$length <- factor(
  as.numeric (
    substr(
      temp[[model]]$length,
      start=3,
      stop=5)))

means <- ddply(temp[[model]],
                .variables = c('length','parameter'),
                'summarise',
                mean = mean(value))

k <- resamp[[1]][[model]]@plot.out$db2[,c("parameter","value")]

best <- ldply(k$parameter, function(j)  {
    db <- subset(means,  subset=(parameter == j))
    true.value <- k$value[k$parameter == j]
    best <- db$length[which.min(abs(true.value-db$summarise))]
    data.frame(length = as.numeric(best),
               parameter = j,
               value = db$summarise[best])
    })
```

```
library(ggplot2)

ggplot(temp[[model]], aes(length,value)) +

  geom_boxplot() +

  geom_hline(aes(yintercept = value), data = k, linetype = "dashed",size = 1) +

  facet_grid(parameter ~., scales = "free") +

  geom_point(data = means, aes(length, summarise, colour = "mean values")) +

  geom_point(data = best,aes(length,value,colour = "best length") ) + labs(colour = "legend"
) +

  scale_colour_hue() +

  ggtitle("Boxplot 100 MBB resamples") +

  theme(plot.title =
          element_text(hjust = 0.5)) +

  theme(legend.position = "bottom",
        legend.title =
          element_text(face = "bold"))
```

In the same fashion we can obtain a table with all the best values:

```
best.table <-  ldply(seq_len(10), function(j) {
  db <- temp[[j]]
  db$length <- factor(
    as.numeric(
      substr(db$length,start=3,stop=5)))

  means <- ddply(db,
                 .variables = c('length','parameter'),
                 'summarise',
                 mean = mean(value))

  k <- resamp[[1]][[j]]@plot.out$db2[,c("parameter","value")]

  best <- ldply(k$parameter, function(j)  {
      db <- subset(means,  subset = (parameter == j))
      true.value <- k$value[k$parameter == j]
      best <- db$length[which.min(abs(true.value-db$summarise))]
      data.frame(length = as.numeric(best),parameter = j)
})
  cbind(best,modelo = j)

})

b.table <- sapply(unique(best.table$modelo), function(j){
  db <- subset(best.table,  subset = (modelo == j))
  sapply(levels(best.table$parameter), function(i) {
    x <- db$length[db$parameter == i]
    if(identical(x, numeric(0))) NA else x
    })
```

```
  })

best.table <- cbind(model = unique(best.table$modelo), t(b.table))

knitr::kable(best.table)
```

HHJ

Now, we proceed to the Hall, Horowitz and Jing (1995) algorithm, as the estimation process is prone to errors, specially with `l = 1:3`, I strongly advise saving the results.

```
names(sims) <- paste("model", 1:10)

loop <- models[models$phi != 0 ,]

for (i in as.numeric(rownames(loop))[4:6]) {
  chosen.model <- paste("model",i)
  ord <- sims[[chosen.model]]@order
  fam <- sims[[chosen.model]]@spec@family
  db <- sims[[chosen.model]]@value[[1]]
  db <- data.frame(yt = db$yt)
  db$x <- 1

  opt <- HHJ(db, dboot::bootf, ord = ord,
             fam = fam,
             export = c("garmaFit2"),
             package = c("gamlss"),
             nsteps = 100,
             type.optm = 2,
             type.sub.blocks = "complete",
             n.try = 200,
             seed = 1010)

  save(opt, file = paste("HHJ_model100_",i,".RData" ,sep=""))
}

  lapply(as.numeric(rownames(loop)), function(j) {
    load(paste("HHJ_model_",j,".RData", sep = ""))
    opt
  })


# now to the plot:

  xx <- lapply(as.numeric(rownames(loop)), function(j) {
    load(paste("HHJ_model_",j,".RData", sep = ""))
    list(tab=table(opt$l.optm),ind=j)})
x <- data.frame()

  for (i in (1:6))  {
    x2 <- data.frame(n.optm=xx[[i]]$tab,model=factor(xx[[i]]$ind))
    x <- rbind(x,x2)
    }
  )

names(x) <- c('L.optim',"Freq.","model")
```

```
x$L.optim <- as.numeric(x$L.optim)

indice <- setdiff(1:10,c(1,2,5,6,7,10))
indice <-(x$model!=indice[1])&(x$model!=indice[2])&(x$model!=indice[3])&(x$model!=indice[4])

library(ggplot2)
ggplot(x[indice,], aes(x=L.optim, y = Freq.,shape=model,colour=model)) +  geom_point() +scal
e_shape_manual(values=c(0,1,2,3,4,6,8,15,16,17)) +
scale_x_continuous(breaks=1:30)+scale_y_continuous(breaks=seq(from=0,to=100,by=10))+
ggtitle(bquote(atop("L optimal of Hall, Horowitz and Jing (1995), 100 iterations for"~phi)))
```

LFL

Now the Lahiri, Furukawa and Lee (2007) algorithm:

```
for (i in as.numeric(rownames(loop))) {
  chosen.model <- paste("model",i)
  ord <- sims[[chosen.model]]@order
  fam <- sims[[chosen.model]]@spec@family
  db  <- sims[[chosen.model]]@value[[1]]

  opt <- LFL(db,bootf,ord = ord,
             fam = fam,
             export = c("garmaFit2"),
             package = c("gamlss"),
             nsteps = 100,
             type.optm = 2)

  save(opt, file = paste("LFL_model_",i,".RData" ,sep=""))

}
```

Given the saved files all there is to be done is to manipulate them and produce the plot:

```
lst <- lapply(as.numeric(rownames(loop)), function (j)  {
  load(paste("LFL_model_",j,".RData" ,sep=""))
  opt$l.adj[2:101]
  })

serie <- lapply(lst, function(j) {
  len <- length(j[[1]])
  db <- matrix(unlist(j),nrow=100, byrow=T)
  db[,2]
  })


x <- lapply(serie, function(i) {
  data.frame(L.optim = unique(i), Freq. =
               sapply(unique(i),
                      function(j) {
                        x <- rep(0,100)
                        x[rep(j,100)==i] <- 1
                        sum(x)
                        }) )
  })
```

```
x <- lapply(1:6, function(j)
  cbind(x[[j]],model = factor((1:10)[models$phi != 0][j])))
x <- ldply(x, function(j)j)

ggplot(x, aes(x=L.optim, y = Freq., shape = model, colour = model)) +
geom_point() +
scale_shape_manual(values = c(0,1,2,3,4,6,8,15,16,17)) +
scale_x_continuous(breaks = 1:30) + scale_y_continuous(breaks = seq(from = 0, to = 100, by
=10)) +
ggtitle(bquote(atop("L optimal of Lahiri, Furukawa and Lee (2007), 100 iterations for" ~ phi
)))
```

REAL DATA ANALYSIS:

The *Poisson* case:

```
#Reading the data:
db <- read.table("Series.txt")
db <- db[1:grep("12/1999", db$data),]

#declaring the time series variable:
bankruptcy <- ts(data = db$falencia, start = 1980, frequency = 12)
bankruptcy

#Basic plots, ACF/PACF:
plot(bankruptcy ,type="p",pch=19,col="red", main="The UCLA-LoPucki Bankruptcy Research Datab
ase",ylab="Number bankruptcy filings ")
par(mfrow=c(2,1))
acf(bankruptcy)
pacf(bankruptcy)
```

The *Poisson* - *GARMA*(1,0) model:

```
library(timeSeries)
library(gamlss.util)

fit <- garmaFit2(bankruptcy ~ intercept-1,
                 data = data.frame(bankruptcy = db$falencia,
                                   intercept = rep(1,nrow(db))),
                 order = c(1,0),
                 family = "PO",
                 tail = 0,
                 control = list(iter.max=1000))
plot(fit)

summary(fit)

residuo <- fit$residuals

#residual plots:
par(mfrow=c(3,1))
hist(residuo, freq=FALSE, xlab="Density", main="Residual", col="orange")
curve(dnorm(x, mean=mean(residuo), sd=sd(residuo)), add=TRUE, col="darkblue", lwd=2)
qqnorm(as.timeSeries(residuo), col="red", pch=19)
qqline(as.timeSeries(residuo))
```

```
plot(y=residuo,x=rownames(db), xlab="Time",ylab = "Residual",  col="red", pch=19, main="Resi
dual X Index ")

par(mfrow=c(2,1))
acf(residuo)
pacf(residuo)
```

The *Poisson* - *GARMA*(2,0) model:

```
set.seed(123)
fit2 <- garmaFit2(bankruptcy ~ intercept-1,
                  data = data.frame(bankruptcy = db$falencia,
                                    intercept = rep(1,nrow(db))),
                  order = c(2,0),
                  family = "PO",
                  tail = 0,
                  control = list(iter.max=1000))
plot(fit2)
summary(fit2)
residuo <- fit2$residuals

#residual plots:
par(mfrow=c(3,1))
hist(residuo, freq=FALSE, xlab="Density", main="Residual", col="orange")
curve(dnorm(x, mean=mean(residuo), sd=sd(residuo)), add=TRUE, col="darkblue", lwd=2)
qqnorm(as.timeSeries(residuo), col="red",pch=19)
qqline(as.timeSeries(residuo))
plot(y=residuo,x=rownames(db), xlab="Tempo",  col="red", pch=19, main="Residual X Index ")

par(mfrow=c(2,1))
acf(residuo)
pacf(residuo)
```

Now the MBB:

```
db2 <- data.frame(bankruptcy = db$falencia,
                  intercept = rep(1,nrow(db)))

# function to bootstrap
bootf <- function (db) {

  fit <- garmaFit2(bankruptcy~intercept-1,
                   data = db,
                   order = c(2,0),
                   family = "PO",
                   tail = 0,
                   control = list(iter.max = 1000))

  return(fit$coef)

  }

nsims <- 1000
lengths <- seq(from = 5, to = 160, by = 10)
```

```
MBB <- lapply(lengths,
              function(j) tsboot2(db2,
                                  statistic = bootf,
                                  R = nsims,
                                  l = j,
                                  packages = "gamlss",
                                  export = 'garmaFit2')
              )



names(MBB) <- paste("l.", lengths, sep = "")

resumo <- t(sapply(seq_len(length(lengths)),
                   function(p)
                     apply(MBB[[p]]$t,
                           2,
                           mean)
                   )
            )

colnames(resumo) <- attr(MBB[[1]]$t0,"names")
resumo <- as.data.frame(resumo)
resumo$"length" <- lengths

#function to bootstrap and return the Gaussian confidence interval:

bootf2 <- function (db) {

  fit7 <-  garmaFit2(bankruptcy ~ intercept-1,
                     data = db,
                     order = c(2,0),
                     family = "PO",
                     tail = 0,
                     control = list(iter.max = 1000)
                     )

  ci <- data.frame( lb = fit7$coef - sqrt(diag(fit7$vcov))*1.96,
                    ub = fit7$coef + sqrt(diag(fit7$vcov))*1.96
                    )
  return(ci)
}

set.seed(123)
norm.teste <- bootf2(db2)
colnames(norm.teste) <- c("norm.LB","norm.UB")

##bootstraped ci's:

conf.interv <- function(data,block.length,par.names) {

  dados <- foreach(j = seq_len(length(par.names))) %:% foreach(i = block.length, .combine =
"rbind") %dopar% {

    norm <- boot:::norm.ci(t0 = data[[i]]$t0[j],
                           t = data[[i]]$t[j,])
```

```
    basic <- boot:::basic.ci(t0 = data[[i]]$t0[j],
                             t = data[[i]]$t[j,])

    perc <- boot:::perc.ci(t = data[[i]]$t[j,])

    dados <- data.frame(conf = norm[1],
                        bias.cor.LB = norm[2], #LB = lower bound
                        bias.cor.UB = norm[3], #UB = upper bound
                        basic.LB = basic[4],
                        basic.UB = basic[5],
                        perc.LB = perc[4],
                        perc.UB = perc[5]
                        )

    rownames(dados) <- i
    dados

    }
  names(dados) <- par.names
  dados
}

intervals <- conf.interv(MBB,
                              block.length = paste("l.",lengths,sep=""),
                              par.names = names(coef(fit2)))

##Manipulating for plotting:
intervals <- lapply(seq_len(length(intervals)), function(j) {

  temp <- rdply(length(lengths),
                norm.teste[j,])[-1]
  rownames(temp) <- NULL

  db <- cbind(conf = intervals[[j]][,1],
              temp,
              intervals[[j]][,-1])

  db$length <- rownames(db)
  db
  }
  )

names(intervals) <- names(coef(fit2))

temp <- ldply(intervals)
names(temp) [1] <- "parameter"
temp <- reshape2::melt(temp)
xx <- reshape2::melt(resumo,id.vars = "length")
names(xx)[2] <- "parameter"
xx$variable <- "mean.value"

x <- temp
x <- reshape2::melt(x)

##Plots:

type.ci <- "bias.cor"  # 'norm', 'bias.cor', 'basic', 'perc'
```

```
temp <- cbind(xx,
              LB = subset(x,
                          variable == paste(type.ci,".LB", sep = ""))$value,
              UB = subset(x,
                          variable == paste(type.ci,".UB", sep = ""))$value)

p2 <- ggplot(temp, aes(x = length, y = value, colour = parameter)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB, x = length)) +
  facet_grid(parameter ~.,scale = "free") +
  ylim(-1,2) +
  ggtitle(paste("1000 MBB resamples:
                Parameter mean values and 95% Bias Corrected CI  estimates")) +
  theme(legend.position = "bottom", plot.title = element_text( hjust = 0.5 ))

type.ci <- "norm"  # 'norm', 'bias.cor', 'basic', 'perc'
temp <- cbind(xx,
              LB = subset(x,
                          variable == paste(type.ci,".LB", sep = ""))$value,
              UB = subset(x,
                          variable == paste(type.ci,".UB", sep = ""))$value)

p1 <- ggplot(temp, aes(x = length, y = value, colour = parameter)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB, x = length))+
  facet_grid(parameter ~., scale="free") +
  ylim(-1,2) +
  ggtitle(paste("1000 MBB resamples:
                Parameter mean values and 95% Gaussian CI  estimates")) +
  theme(legend.position = "none", plot.title = element_text( hjust = 0.5 ))

library(gridExtra)
grid.arrange(p1,p2,ncol=1)
```

HHJ:

```
bootf3 <-function (data, ord, fam) {

  fit7 <- garmaFit2(formula(bankruptcy ~ intercept - 1),
                    data = data,
                    order = ord,
                    family = fam,
                    tail = 0,
                    control = list( iter.max = 1000 )
                    )
  return(fit7$coef)
  }

ord <-  c(2,0)
fam <- "PO"

algo2_bias <- HHJ(db2,
                  bootf3,
                  R = 100,
                  nsteps = 10,
                  ord = ord,
```

```
                  fam = fam,
                  export = c("garmaFit2"),
                  packages = c("gamlss"),
                  m.init = 30)

algo2_ci   <- HHJ(db2,
                  bootf3,
                  R = 100,
                  nsteps = 10,
                  ord = ord,
                  fam = fam,
                  export = c("garmaFit2"),
                  packages = c("gamlss"),
                  m.init = 30,
                  type.est = 'two.sided.distribution')

hhj       <-  HHJ(db2,
                  bootf3,
                  R = 100,
                  nsteps = 10,
                  ord = ord,
                  fam = fam,
                  export = c("garmaFit2"),
                  packages = c("gamlss"),
                  m.init = 30,
                  type.est = "two.sided.distribution",
                  type.optm = 1,                          ####optimizing for the intercept te
rm
                  type.sub.blocks = "complete"            #### perform the HHJ algorithm
                  )
```

The *Gamma* case:

```
#Reading the data:
load("quotes.RData")
tik <- quotes[["BPHA3.SA"]]
y <-  pmax(diff(log(tik))^2,1e-4)*1e+4

#declaring the time series variable:
log.return <- as.ts(as.vector(y))

#Basic plots, ACF/PACF:
par(mfrow = c(2,1))
acf(log.return)
pacf(log.return)

#ARCH and Ljung-Box tests:

lags <- c(1,4,8,12,24)
alpha <- 0.1
len <- c(30,60,90,180,360,504)

foreach (n =  1, .packages = c("FinTS")) %:% foreach(lags = lags,.combine="rbind") %dopar%{
  a <- y
  arch <- ArchTest(a, lags)$p.value <= alpha                                    ### H0: n
o Arch effect
```

```
  ljung.box <- Box.test(a^2, lag = lags,type = c("Ljung-Box"))$p.value <= alpha    ### H0: n
o autocorrelation different of zero
  db <- data.frame(lag = lags, arch = arch, ljung.box = ljung.box)
  rownames(db) <- NULL
  db                                                                               ### data.
frame with H0 rejection = TRUE/FALSE for arch and ljung.box
 }
```

The *Gamma* - *GARMA*(1,0) model:

```
source("garmagarchFit.R")
library(gamlss.util)
library(timeSeries)

set.seed(1234)
fit.garch <- garmagarchFit(yt ~ .,
                           order = c(1,0),
                           data = data.frame(yt = tail(y,180)),
                           family = "GA")

summary(fit.garch)

residuo <- fit.garch$residuals

#Model parameter estimation stability:

library('doRNG')

set.seed(123)
mc.gamma <- foreach(i = seq_len(1000), .packages = c("gamlss.util")) %dorng% {

  fit.gamgarch <- garmagarchFit(yt ~ .,
                                order = c(1,0),
                                data = data.frame(yt = tail(y,180)),
                                family ="GA")
  return(list(param = fit.gamgarch$coef, aic = fit.gamgarch$aic))
  }

library(plyr)

mc.gamma <- ldply(mc.gamma,.fun = function(j) c(j$param,j$aic))
names(mc.gamma)[4] <- "AIC"
mc.gamma<- reshape2::melt(mc.gamma)

p1 <- ggplot(subset(mc.gamma,variable ==  "beta.(Intercept)")) +
  aes(value)+
  geom_density(alpha = 0.2) +
  ggtitle(paste("Histogram 1000 fits", "beta.(Intercept)"))

p2 <- ggplot(subset(mc.gamma,variable == "phi" ), aes(value)) +
  geom_density(alpha = 0.2) +
  ggtitle(paste("Histogram 1000 fits","phi")) +
  theme(legend.position = "bottom", plot.title = element_text( hjust = 0.5 ))

p3 <- ggplot(subset(mc.gamma,variable == "AIC"), aes(value, fill = model)) +
  geom_density(alpha = 0.2) +
```

```
  ggtitle(paste("Histogram 1000 fits","AIC"))

stats <- ddply(mc.gamma,
               .variable=c('variable'),
               "summarise",
               Median = median(value),
               Mean = mean(value),
               Min= min(value),
               Max=max(value))

stats <- reshape2::melt(stats)

stats$value <- round(stats$value,4)

#residual plots:

par(mfrow = c(3,1))
hist(residuo, freq = FALSE, xlab = "Density", main = "Residual", col = "orange")
curve(dnorm(x, mean = mean(residuo), sd = sd(residuo)), add = TRUE, col = "darkblue", lwd =
2)
qqnorm(as.timeSeries(residuo), col = "red", pch = 19)
qqline(as.timeSeries(residuo))
plot(y = residuo, x = seq_len(length(residuo)), xlab = "Tempo",  col = "red", pch = 19, main
 = "Residual X Index")

par(mfrow = c(2,1))
acf(residuo)
pacf(residuo)

#Arch Test & Ljung-Box for the residuals:

foreach (n =  1, .packages = c("FinTS")) %:% foreach(lags = lags,.combine="rbind") %dopar%{
  a <- residuo
  arch <- ArchTest(a, lags)$p.value <= alpha                                    ### H0: n
o Arch effect
  ljung.box <- Box.test(a^2, lag = lags,type = c("Ljung-Box"))$p.value <= alpha    ### H0: n
o autocorrelation different of zero
  db <- data.frame(lag = lags, arch = arch, ljung.box = ljung.box)
  rownames(db) <- NULL
  db                                                                            ### data.
frame with H0 rejection = TRUE/FALSE for arch and ljung.box
 }
```

Now the MBB:

```
#function to bootstrap:
bootf <- function (db) {
  fit <- garmagarchFit(formula(yt ~ x - 1),
                       data = db,
                       order = c(1,0),
                       family = "GA",
                       tail = 0,
                       control = list( iter.max = 1000)
  )
  return(fit$coef)
  }
```

```r
db3 <- data.frame(yt = tail(y, 180))
db3$x <- 1

nsims <- 1000

lengths <- sort(c(12, seq( from = 5, to = 60, by = 5)))[-c(6,9,10,12,13)]

set.seed(123)

MBB <- lapply(lengths, function(j) {
  print(j)

  tsboot2(db3,
          statistic = bootf,
          R = nsims,
          l = j,
          packages = "gamlss",
          export = 'garmagarchFit')
  }
  )

resumo <- t(sapply(seq_len(length(lengths)), function(p) apply(MBB[[p]]$t, 2, mean)))
names(MBB) <- paste("l.",lengths, sep = "")

tsboot2(db3, statistic = bootf, R = nsims, l = 10, packages = "gamlss", export = 'garmagarch
Fit')

resumo <- t(sapply(seq_len(length(lengths)),function(p) apply(MBB[[p]]$t,2,mean)))
colnames(resumo) <- attr(MBB[[1]]$t0,"names")
resumo <- as.data.frame(resumo)
resumo$"length" <- lengths
names(resumo)[1] <- "Intercept"

#function to bootstrap and return the Gaussian confidence interval:
bootf2 <- function (db) {

  fit <- garmagarchFit(formula( yt ~ x - 1),
                       data = db,
                       order = c(1,0),
                       family = "GA",
                       tail = 0,
                       control = list( iter.max = 1000))

  ci <- data.frame( lb=fit$coef - sqrt(diag(fit$vcov))*1.96,
                    ub=fit$coef + sqrt(diag(fit$vcov))*1.96)
return(ci)
}

set.seed(123)
norm.teste <- bootf2(db3)
colnames(norm.teste) <- c("norm.LB", "norm.UB")

intervals <- conf.interv(MBB,
                         block.length = paste("l.",lengths,sep=""),
                         par.names = c(names(coef(fit.teste)), names(fit.teste$sigma)))
```

```
intervals <- lapply(seq_len(length(intervals)), function(j) {
  temp <- rdply(length(lengths), norm.teste[j,])[-1]
  rownames(temp) <- NULL
  db <- cbind(conf = intervals[[j]][,1], temp,intervals[[j]][,-1])
  db$length <- rownames(db)
  db
})
names(intervals) <- c(names(coef(fit.garch)),names(fit.garch$sigma))

temp <- ldply(intervals)
names(temp) [1] <- "parameter"
temp <- reshape2::melt(temp)
x <- temp
x <- reshape2::melt(x)
xx <- reshape2::melt(resumo, id.vars = "length")
names(xx)[2] <- "parameter"
xx$variable <- "mean.value"
xx$tag <- "normal"
xx$tag[xx$length == 12] <- "Algorithm 2"

type.ci <- 'bias.cor'  # 'norm', 'bias.cor', 'basic', 'perc'
temp <- cbind(xx,
              LB = subset(x, variable == paste(type.ci, ".LB", sep=""))$value,
              UB = subset(x, variable == paste(type.ci, ".UB", sep=""))$value)

p2 <- ggplot(temp, aes(x = length, y = value, colour=parameter)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB, x = length))+
  facet_grid(parameter ~.,scale = "fixed") +
ggtitle(paste("1000 MBB resamples:
 Parameter mean values and 95% Bias Corrected CI  estimates")) +
  theme(legend.position = "bottom", plot.title = element_text( hjust = 0.5))

type.ci <- 'norm'  # 'norm', 'bias.cor', 'basic', 'perc'
temp <- cbind(xx,
              LB = subset(x, variable == paste(type.ci, ".LB", sep=""))$value,
              UB = subset(x, variable == paste(type.ci, ".UB", sep=""))$value)

p1 <- ggplot(temp, aes(x = length, y = value, colour=parameter)) +
  geom_point() +
  geom_errorbar(aes(ymin=LB, ymax=UB, x= length)) +
  ylim(-3,5)+
  facet_grid(parameter ~., scale = "fixed") +
ggtitle(paste("1000 MBB resamples:
 Parameter mean values and 95% Gaussian CI  estimates")) +
  theme(legend.position = "none", plot.title  =element_text( hjust= 0.5))

library(gridExtra)
grid.arrange(p1,p2,ncol=1)
```

HHJ:

```
bootf3 <-function (data, ord, fam) {

  fit <- garmagarchFit(formula( yt~ x - 1),
                       data = data,
```

```
                        order = ord,
                        family = fam,
                        tail = 0,
                        control = list( iter.max = 1000 ))

    return(fit$coef)
  }

ord <-  c(1,0)
fam <- "GA"

algo2_bias <- HHJ(db3,
                  bootf3,
                  R = 100,
                  nsteps = 10,
                  ord = ord,
                  fam = fam,
                  export = c("garmagarchFit"),
                  packages = c("gamlss"),
                  m.init = 10)

algo2_ci <- HHJ(db3,
                bootf3,
                R = 100,
                nsteps = 10,
                ord = ord,
                fam = fam,
                export = c("garmagarchFit"),
                packages = c("gamlss"),
                m.init = 10,
                type.est = 'two.sided.distribution')

hhj_intercept <- HHJ(db3,
                     bootf3,
                     R = 100,
                     nsteps = 10,
                     ord = ord,
                     fam = fam,
                     export = c("garmagarchFit"),
                     packages = c("gamlss"),
                     m.init = 10,
                     type.est = "two.sided.distribution",
                     type.optm = 1, #1,2,3
                     type.sub.blocks = "complete")
```

Final Step:

As we have finished working with a parallel environment it is a good practice to finish the parallel workers if we will keep the current R session:

```
doParallel::stopImplicitCluster()
```