Sapienza
Università di Roma

Exploring Conformal Prediction in Long Memory Processes

Facoltà di Economia
Laurea Magistrale in Financial Risk and Data Analysis

Alessandra Campanella
ID number 1944365

Advisor
Prof. Deliu Nina

Academic Year 2024/2025

Exploring Conformal Prediction in Long Memory Processes
Tesi di Laurea Magistrale. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: alessandra.campanella19@gmail.com

Indice

Capitolo 1

Introduction

In this thesis, I will analyze long memory behavior in time series, focusing on applying conformal prediction to construct prediction intervals.

Looking at statistical modeling and time series analysis, it is usually assumed that the observations made are independent from each other, or at least the connection is very weak, the most used statistical methods are built on specific models, such as the Gaussian white noise and Autoregressive Moving Average (ARMA) processes. The use of these models suggests that the links between observations become weaker quickly, we can think about this concept in everyday terms: recent events often have a bigger impact on the present than those that happened a long time ago.

The various methodologies we develop can be applied under the assumption that the relationship between distant observations weakens rapidly. This implies that, for practical purposes, observations from the distant past become increasingly less significant. While this exponential decay assumption simplifies mathematical analysis, it is also evident in daily experiences, as recent events typically hold more sway over current outcomes than those from long ago.

The discovery of long-memory processes creates a fundamental crisis for traditional methods such as the maximum likelihood for parameter estimation, since even very distant observations continue to influence current values in a meaningful way.

The differences discovered by applying long-memory are various. First, the standard estimation methods, which work well under weak dependence assumptions, will produce inconsistent results when applied to long-memory data since the method will not give us a true estimate; another difference is that confidence intervals will not be based on the assumption that correlations disappear quickly, but will cover different probabilities. Also, it can be noticed that forecasting methods designed for traditional time series can underestimate uncertainty in predictions for long-memory processes.

This work is structured as follows.

In Chapter 2 I introduce long memory processes and the mathematical framework behind them, which describes what happens if correlation persists much longer than the traditional model expects. Long memory processes, unlike traditional time series processes, describe situations where the influence of past observations decays slowly over time. In traditional models, past shocks can impact current results but have little to no effect on future ones. We begin with a basic definition and identify the key parameter that controls the strength of long-memory behavior, next,explore the most important models used to describe long-memory processes, focusing on fractional ARIMA models. These models introduce a fractional parameter, d, which governs long-memory effects. The statistical properties of these new processes are then examined, to highlight how they differ from traditional time series. This includes an understanding of why sample means behave differently, why confidence intervals exhibit unexpected characteristics, and why forecasts may have unusual properties. Finally, I discuss practical implications, including how to recognize long-memory behavior in real data and which estimation methods to use.

This chapter is directly related to our main research question. By examining how long-memory processes challenge traditional statistical methods, we can understand why conformal prediction may be a viable solution. The ongoing correlations that make regular prediction intervals unreliable might not pose the same problems for conformal prediction, which is based on much simpler assumptions. Thorough this analysis it has been established the foundation necessary to carefully evaluate how well conformal prediction methods perform with long-memory data. It is crucial to thoroughly understand these processes to develop effective solutions to the challenges they present.

To work effectively with long-memory processes, we need to employ concrete models that explain how long memory manifests in real data. This analysis is divided into four subsections, each focusing on a different model. These models will help us understand the mathematical framework necessary for making predictions in long-memory processes. Each model provides unique insights into how persistent correlations can emerge from seemingly simple underlying processes.

The first model we analyze is the aggregation of short-memory processes, which offers a clear explanation of the behavior of long-memory processes and why they frequently appear in real-world data. The aggregation model demonstrates how combining different short-memory processes can result in complex long-range dependence.

This model has quite a high significance. It links the simple behaviors of individual parts to the bigger patterns in the whole system. This specific connection is fundamental to better understand how long memory processes work in fields such as economics, finance, and climate science, where different actions contribute to overall trends.

From a practical point of view, this model encourages the use of specific mathematical tools like ARFIMA models to work with long-memory data, even if the real-life process behind it does not exactly match the math.

The second model we will examine is the Ising model, a fundamental statistical mechanics concept. This model suggests that long-lasting memory can be understood by looking at physical principles. By focusing on the Ising model, we can notice how, rather than thinking about correlation persistence as something happening only on time series data, this model shows how these long-lasting schemes build up naturally when they get close to critical points. This perspective is really interesting since it connects long memory processes to some universal physics principles which can be applied to different systems.It is useful because it shows us when and why long memory can be particularly strong or last a long time. Systems that are close to critical points are exactly the ones where regular prediction methods often don't work well. This makes them perfect for trying out other methods, like conformal prediction.

The third model studied is the Hierarchical Variation Model. This model helps us understand hierarchical variation and how it generates long memory. An important characteristic of this model is that random fluctuations are introduced within a structured process. We see examples of hierarchical structures everywhere in our everyday life, such as production lines in factories where they have different control levels. What makes this model an object of our studies is that, unlike other models that just look at how similar processes can be combined, the hierarchical model is relevant for any process with multiple stages where randomness appears at different levels. The main feauture is that expresses long memory in processes called hierarchical variation, where the way real-world processes are organized can lead to long-term dependencies, also the way it reflects how complex systems are usually structured makes a big difference from other aggregation models, since mixes multiple processes running at the same time. The Ising model on the contrary needs systems to be near critical points, this is why this model is used to fit layered and multi-step nature such as in factory operations, it takes its origins from Cox and Townsend's work in 1947 due to issues in the textile industry, which is why this model is primarily used to tackle real-world issues.Theoretically, explains that the structure is just as important as the individual parts, and as told from the practical example above, the way sources or systems are organized at different levels can lead to long-term memory in the whole system. It also links the randomness at the micro level to the long-term stability at the macro level, while each level in the hierarchy may be easy to understand on its own, their interaction produces new properties that differ from those of any individual part.Analyzing long memory in this type of model is crucial for understanding the predicted outcomes. If the long memory originates from the model, our prediction methods must take into account the influences that occur over different time frames. Various levels of hierarchy may significantly impact the results at different scales. As we investigate conformal prediction in the context of long-memory processes, we encounter both challenges and opportunities presented by the hierarchical model. Traditional prediction methods may not only struggle due to long-range correlations but also fail because they overlook the complexities of influences that occur at various scales. The distribution-free nature of conformal prediction is particularly advantageous in this scenario, as it does not require us to precisely define the intricate hierarchical structure. Additionally, the model suggests that prediction success may change predictably over different time frames, influenced by the hierarchical setup. Understanding these trends could help us identify when and why conformal prediction performs better than traditional methods.

The fourth and last model is the one based on the stochastic partial differential equations. This one is based on different equations that help us understand how long-memory dependence comes about in natural processes. This allows us to look at long-range dependence not just in time but also in space, this is fundamental when applied to real-life situations such as environmental changes or looking at financial markets in specific locations. This last model is strongly related to basic physical ideas. Whittle from Beran (2017), back in the 1950s and 1960s, created this model to explain how long memory coming from basic physics equations can explain how things spread out in time and space. The SPDE framework offers both benefits and challenges for exploring conformal prediction methods. On one side, the model is based on physical principles, which gives us a solid reason to expect long-memory effects. This makes it a great place to test how different prediction methods manage long-range dependence that's driven by physical factors.On the other hand, it can be complex and may require us to estimate spatial parameters that are difficult to determine from limited data. However, the distribution-free approach of conformal prediction could be beneficial in this context, as it does not require us to accurately define and estimate intricate spatial correlation patterns.

The third chapter is about Conformal prediction, which can be distinguished into Standard and Adaptive CP, each of which is applied in a specific context based on the hypothesis of exchangeability or on the presence of shifting distributions over time.

The Conformal Prediction's underlying idea is of building predictive confidence intervals with a guaranteed coverage, without asking strong hypotheses about the error distribution or the model's functional form. This makes Conformal Prediction particularly suited for the analysis of historical time series influenced by long-term dependencies, for which classical parametric methods are often inadequate because they are based on assumptions of independence or weak dependence.In the initial part of the chapter, the theoretical framework of the standard conformal prediction will be illustrated, based on the exchangeability assumption of the observed time series.

Then the main implementation procedures will be formalized, divided into Split Conformal Prediction and Full Conformal, with a focus on the conformity scores construction, critical quantile estimation, and the coverage level interpretation. Specifically, we will analyze how to obtain robust predictions even with arbitrary predictive models, through the use of calibration sets different from the training ones.

Afterwards, we will move on to the Adaptive Conformal Prediction, developed to deal with the limitations imposed by exchangeability in real-world settings.

The ACP was introduced by Gibbs and Candes (2021) and extended by Zaffran et al. (2022). It is used to dynamically deal with the variation of the data distribution, which is typical behavior of financial time series. This method proceeds by adapting the quantile threshold and the miscoverage sequence, using an online updating and monitoring procedure for the errors. These characteristics make the ACP particularly suited for structures with complex serial dependencies, such as the ones from Chapter 2. The importance of selecting the right base learners will also be discussed. Although Conformal Prediction is model-free concerning the error distribution, the number of intervals created will strongly depend on the accuracy of the point forecasts. In this context, ARFIMA models are ideal; the use of the fractional differencing parameter d allows for modeling the persistence effect typical of long memory processes, ensuring good stationarity for a value of $d<0.5$ and an efficient calibration on residual errors.

This chapter seeks to connect long-memory process theory with effective predictive techniques, showcasing how conformal prediction serves as a practical and sound theoretical option. We will outline criteria for model selection, explore the computational implications of various approaches, and compare these with traditional parametric methods. The theoretical insights presented will lay the groundwork for the empirical and simulation analysis in Chapter 4.

Chapter 4 aims to apply the conformal prediction methods described in the previous chapter to long-memory time series. Once the theoretical nature of conformal prediction and ARFIMA model has been explored, we now aim to analyze the validity and effectiveness of these tools through computational simulation and applications on real data. The initial part is about the ARFIMA and ARIMA model specification, with a focus on the importance of the fractional differencing parameter d, which determines the depth of memory in the process. Then the stationary criteria will be illustrated, together with the spectral properties and the provisional implications connected to long-range dependencies. The discussion will also include the parameter estimation methods, such as the maximum likelihood approach and the Whittle method, focusing also on their empirical implementation. Next, a simulation study aimed at testing the performance of conformal prediction applied to series generated by ARFIMA processes will be developed. The goal is to evaluate empirically the predictive intervals coverage and the average interval width, focusing also on how the parameter d can influence these metrics. Lastly, the chapter will conclude with two application cases on real data: the time series analysis of the WTI oil price and the SP 500 stock index. These two datasets were chosen for the presence of long-memory behavior, highlighting their significance in contexts where quantifying predictive uncertainty is essential for economic and financial decision-making. We will compare intervals constructed through conditional probability (CP) with those derived from traditional parametric methods, assessing the trade-offs between statistical robustness, data fit, and computational complexity.

Capitolo 2

Long-memory Processes

2.1 Theory of Long-memory Processes

Long memory processes describe time series or stochastic processes where correlations between observations decay so slowly over time that their sum diverges. This stands in contrast to short-memory processes like ARMA models, where the autocorrelation function decays exponentially and becomes insignificant after just a few lags.

In mathematical terms, a process X_t is characterized by long memory if its autocorrelation function $\rho(k)$ decreases at a hyperbolic rate, expressed as $\rho(k) \sim Ck^{-\beta}$ where $0 < \beta < 1$ where k represent the number of lags.This results in a divergent sum: $\delta(\rho) = \sum_{k=-\infty}^{\infty} |\rho(k)|$ which indicates that observations separated by long periods in time will still exhibit a meaningful correlation, here the decay is so slow that even distant past observations significantly influence the present, and this persistence has several deep consequences.The first consequence is that the variance of the sample mean under long memory decreases more slowly than the classical $1/n$ rate. In fact, for a long-memory process, the variance of the sample mean behaves as

$$Var(\bar{X}) \sim cn^{-\alpha} \; with \; 0 < \alpha < 1.$$

Secondly, long memory implies that the central limit theorem might not work as usual. This affects theoretical statistics and areas where we see long-range connections, such as finance, water studies, climate science, and analyzing network traffic. In these situations, traditional methods might underestimate risks or get caught up in short-term changes.

At the beginning of his study about long memory, Beran (2017) revisits the concept of variance of the sample mean quoted above, that is, the variance of the sample mean is σ^2/n under independence.He dismantles this assumption, stating that this result cannot be achieved under long-range dependence, so when observations are serially correlated, to prove this, he quantifies the correction through the application of an auto-regressive model, specifically an $AR(1)$ case:

$X_t = aX_{t-1} + \epsilon_t$ where $a \in (-1, 1)$

with normally distributed independent increments ϵ_i with zero mean and variance σ^2;

The autocorrelation function of this process is given by:

$\rho(k) = a^k$

By entering this into the formula for the variance of the sample mean under dependence, we obtain:

$\mathrm{Var}(\bar{X}_n) = \frac{\sigma^2}{n} \left(1 + 2\sum_{k=1}^{n-1} \left(1 - \frac{k}{n}\right) \rho(k)\right)$

The term $c(a)$ acts as a correction factor, so when a is close to 1, the correction is substantial.

Similarly, if the value of a is close to -1, the observations will not tend to assume similar values. For example:

From Table 2.1, following Beran (2017),it is shown that using the variance formula σ^2/n without an appropriate correction can lead to a significant underestimate of the variance of the sample, with a possible correction factor of up to 19 when the parameter a is set to 0.9.

a	c(a)
0.2	1.5
0.6	4.0
0.9	19.0

Tabella 2.1. Correlation a-c(a)

This underestimation carries substantial implications, since the confidence interval may be excessively narrow, and this will cause the validity of the hypothesis test to be compromised, and the forecast could fail to accurately reflect the degree of uncertainty involved.

Beran (2017) then generalizes these assumptions to any stationary process, defining:

$\delta(p) = \sum_{k=1}^{\infty} p(k),$

So that:

$Var(\bar{X}n = \frac{\sigma^2}{n}(1 + 2\delta(p))$

where, in long-memory contexts, $\delta(p) = \infty$.

In these cases, the variance of the sample mean decays more slowly, such as:

$Var(\bar{X}n) \sim \frac{1}{n^{\alpha}}$ with $\alpha < 1$

The alteration in this rate is what fundamentally differentiates long-memory processes and underlines their significance.

The concept of long memory processes is fundamentally differentiated by their unique rate of decay, which underscores their importance.

It is also essential to remember the long memory about the Hurst effect, originally documented by Hurst in Baillie (1996) hydrological studies. In his analysis, Hurst examined 900 different datasets, Baillie (1996), and found the value of the parameter to be 0.73, suggesting the prevalence of long memory in such data.

According to Baillie (1996), the Hurst coefficient can be estimated using the Rescaled Range (R/S) statistic. This method relies on:

$\log[E(R_T/S_T)] \approx \text{constant} + H[\log(T)]$

Where:

- R_T is the range
- S_T is the sample standard deviation
- T is the time sample

Then, according to Baillie (1996) we can use two different approaches to estimate H:

The first approach is the direct one, where H is estimated through $\hat{H} = S_T \log(T)$, or it can be expressed using a regression approach by taking the slope coefficient of a regression $\log[R_t/S_t] = \alpha + H\log(t) + \epsilon_t$ for different values of t.

Baillie (1996) then classified the value of H based on the fractional differencing parameter d, which is related to H as $d = H - 1/2$. Classifying in this way also helps us classify different time series:

- If $H = 0.5$, then $d = 0$, indicating a short memory process where correlations decay exponentially;
- A value of $0.5 < H < 1$ with $0 < d < 0.5$, suggests long memory, with a hyperbolic decay of correlations;
- $H = 1$ and $d = 0.5$ represent the limit of stationarity;

2.2 Examples Of Long-memory Models

Model 1: Aggregation Of Short-memory Processes

One of the most widely studied explanations for the appearance of long memory in observed time series is the aggregation of short-memory processes. In 1980, Clive Grange through Beran (2017)introduced a concept that highlights how combining various independent or weakly dependent processes can result in a new process exhibiting long memory characteristics.

Specifically, under certain conditions, the aggregated process may demonstrate independence, even if each component only shows short-range dependence.This insight is not only mathematically interesting but also practically important, as many real-world systems are made up of multiple interacting components. If we suppose that we have a large number of independent AR(1) processes indexed by j:

$X_t^j = a_j X_{t-1}^{(j)} + \epsilon_t^{(j)} \; with \; -1 < |a_j| < 1$

Then let's consider the aggregate process:

$X_t = \sum_{j=1}^{N} X_t^{(j)}$

If the values of a_j are drawn from a distribution that has, for example, many units near a unit root, then this will tell us that the aggregate process Xt can exhibit long memory. As a consequence, if the process exhibits long memory, then the aggregate autocorrelation function decays at a hyperbolic rate:

$\rho(k) \sim Ck^{-\beta}$ with $0 < \beta < 1$

And hence the sum of autocorrelation diverges:

$\sum_{k=0}^{\infty} \rho(k) = \infty$

This result is aligned with the definition of long memory, where shocks' influence persists for long time horizons. The basic idea is that even if every single $AR(1)$ process is not influenced by its past, the whole ensemble is not. In particular, when the persistence parameters are distributed in a heterogeneous way and clustered near the value of 1, then even small levels of persistence at the individual level can aggregate into substantial persistence at the system level. This concept is particularly relevant in macroeconomics, where aggregate data such as GDP and inflation result from the actions of many individual agents. Each agent might be described by a stable AR(1) process, but their combined behavior can generate an output that exhibits long-term dependence. Similarly, in climatology, aggregated data from multiple regions or components (each with short memory) may produce long-memory features at the global level. This aggregation effect has also been documented in finance, where returns from portfolios or indices reflect the combined behavior of individual assets. From a mathematical perspective, this model is powerful because it does not require any single process to exhibit long memory; the phenomenon emerges from the distribution of the parameters.

In fact, under mild regularity conditions, a fractionally integrated process can approximate the resulting aggregated process, or $ARFIMA$ (AutoRegressive Fractionally Integrated Moving Average) model. The fractional differencing parameter is directly linked to the strength of long memory in time series data. However, there are practical challenges to consider. The aggregation model serves more as an explanation than as a method for estimation. While it does support the use of $ARFIMA$ models, it does not provide a direct method for estimating their parameters.

The precise distribution of these parameters remains elusive, complicating the empirical calibration process for the model. Nonetheless, the aggregation model remains a fundamental theoretical foundation for long memory and has profoundly impacted the creation of applied models, including $ARFIMA$.

Model 2: Ising Model

Another explanation for long memory is given by statistical physics, in particular, it derives from the study of critical points. In this kind of system, variables interact over long-range distances or extended periods, the result is a power-law decay of correlations, which differs from the exponential decay seen in short-memory processes. An example of this type of model is the Ising Model from statistical mechanics. In this model, binary variables called spins are arranged in a lattice structure and interact with their neighboring spins, Beran (2017).

If we consider a large, finite d-dimensional space, each point in this space is associated with a sign, denoted by $S_i \in \{+1, -1\}$.

In the model, each spin interacts with the other in the neighborhood through local forces, which is why spins tend to go in the same direction. The energy of a spin configuration s_i is given by:

$E = -J \sum_{i,j} s_i s_j$

where s_i and s_j are the spins in the neighborhood, and $J > 0$ is the interaction strength. The total magnetization is given by:

$M_A(S) = \sum_{i \in A} S_i$

Under normal conditions, where spins are only weakly dependent, the variance of the magnetization is proportional to $|A|^{-1}$ and is:

$Var(M_A(S)) \sim c|A|^{-1}$

This model shows that at critical temperature, correlation decays slowly and shows us strong long-range dependence and large blocks os spins stay correlated. It can be asserted that by observing magnetization over time, the autocorrelation will decay hyperbolically rather than exponentially, and as a result, the spectral density will diverge at low frequency:

$f(\lambda) \sim \lambda^{-\gamma}$ as $\lambda \to 0$, marking long memory.

This framework can be observed in fluid turbulence, earthquake patterns, and market crashes since it establishes a connection between long memory and universal physics.

Model 3: Hierarchical Variation Model

An interesting explanation for the occurrence of long memory, in particular in industrial and natural environments, is given by the concept of hierarchical variation. This notion was examined by Cox and Townsend in 1947 about manufacturing, with a specific focus on the textile industry. The hierarchical structure of processes, where random perturbations are introduced at different scales, naturally leads to long-range dependence.

To explain the hierarchical structure, we have to consider a manufacturing process, such as yarn production, where the material undergoes multiple stages of transformation. At every stage, new random fluctuations are introduced. We denote the measurement unit with $X^{(N)}(t)$, the model obtained is:

$X^{(N)}(t) = \sum_{s=0}^{N} a^{N-s} U(ta^{-(N-s)}),$

Where:
- $a \in (0,1)$ is the scaling factor
-$U^{(s)}(t)$ are independent noise processes at each hierarchical level. Even though each noise process may demonstrate only short-range dependence, combining these processes across various scales can lead to long memory effects. While analyzing the covariance function between two points, an hyperbolic decay can be observed. This indicates that the correlations between distant observations diminish much more gradually than in standard short-memory models, thereby meeting the criteria for long memory. This allows to say that this kind of process preserves the memory of the previous random shocks across a wide range of scales and also that the hierarchical variation model can be applied in areas beyond textile manufacturing. This model is particularly effective because it does not necessitate that any single source of noise possesses long memory. Rather, it is the fundamental structure of

the process, defined by various layers of independently introduced randomness, that generates long-range dependence.

There are practical challenges to consider. It can be hard to estimate key factors like the number of levels in a hierarchy, how much each level decreases influence, and the characteristics of noise sources based on actual data.

It can be hard to differentiate whether long memory comes from a hierarchy or other reasons, like aggregation or critical phenomena. Even with these difficulties, the hierarchical variation model is still an important tool for understanding and analyzing long memory in different areas. This model highlights how the structure of a process matters a lot more than just how individual parts behave when it comes to the statistical features of the data we see.

Model 4: SPDE

To explain long memory, in particular in spatial data, stochastic partial differential equations (SPDEs) have been proposed by Whittle (1956,1962) in Beran (2017). This framework helps understand how a spatial process changes when random factors come into play. It shows how a gradual loss of connections can be explained by the laws of nature. Let's take an example of something like soil fertility, which changes as we look at it over different areas and times.

This quantity is represented by a stochastic process Y_v with $v = (x,t) = (x_1, ..., x_m, t) m \geq 1$;

The evolution of this process Y_v is governed by the stochastic partial differential equation:

$\frac{dY_v}{dt} = V^2 Y_v + \epsilon_v$

Where $V^2 = \frac{d^2}{dx_1^2} + \frac{d^2}{dx_2^2} + ... + \frac{d^2}{dx_m^2}$ is the Laplacian operator that represents spatial diffusion. The Laplacian operator illustrates the process by which materials go from regions of higher concentration to those of lower concentration. This phenomenon is observed in different physical systems, including heat flow and chemical dispersion.

Solving the SPDE provides interesting insights about the behavior of the spatial covariance function between two locations x and x', which can be seen as follows:

$cov(Y_{x,t}, Y_{x',t}) = \gamma(||x - x'||) \sim c||x - x'||^{2-m}$

Where m represents the spatial dimension. In the context of a two-dimensional space, $m = 2$, we can notice that the decay of correlations occurs quite slowly. This gradual decay shows that there is a long-lasting influence in the spatial domain. Unlike traditional models that often treat distant observations as independent, in this case, we see that the influence remains strong even over large distances.

Capitolo 3

Conformal Prediction

Modeling time series data that shows long-term trends can be quite tricky, especially in fields like climate science and hydrology, where making accurate predictions matters. One example talked about by Beran (2017) looks at the low points of the Nile River. They share how tough it is to balance theories with real-world accuracy when working with long-term patterns that regular models often have a hard time capturing.

The introduction of Adaptive Conformal Inference (ACI) by Zaffran et al. (2022) in their paper "Adaptive Conformal Predictions for Time Series" provides a flexible way to generate accurate prediction intervals without depending on specific distributions. This distribution-free approach is especially helpful when it is hard or complicated to define the error distribution. The success of ACI relies on the chosen predictive model's ability to understand time dynamics found in long-memory processes. Among the different models explored in depth in the previous chapter, the most compatible with ACI turns out to be an aggregation of short-memory processes, for different theoretical and practical reasons.

First of all, it is good to specify that ACI requires a basic model for forecasting that consists of a well-defined one-step-ahead prediction and some residuals whose distribution can be calibrated. The aggregation framework provides exactly these characteristics; in fact, it creates models like $ARFIMA(p, d, q)$, which keep the statistical patterns of ARMA models while adding a fractional integration d parameter to handle persistence (Beran (2017)). This helps produce manageable residuals $r_{t+1} = X_{t+1} - \hat{X}_{t+1}$ and clear one-step-ahead forecasts $\hat{X}_{t+1}$, which are important for using ACI effectively, in particular in the calibration step.

The second reason is that aggregation models differ from those focused on critical phenomena or stochastic PDEs because they maintain stationarity when $d < 0.5$, ensuring that residuals do not drift out calibration range. In addition, $ARFIMA$ models come with straightforward estimation methods, such as the

Whittle Semiparametric Estimation, the Maximum Likelihood, or the Bayesian method. This feature is important when you need to adjust your model based on residuals in real-time and under computational constraints. As empirical evidence, it can point out that climatic processes come from the combination of many small subsystems, like local water systems or regional weather patterns; each of these subsystems can be described using a simple statistical model, such as the $AR(1)$ or the $ARMA$.

In his work, Beran (2017) shows that when you combine these $AR(1)$ processes, especially those with coefficients close to one, you end up with a pattern that behaves differently over time and needs a special approach called fractional integration to understand it better. This makes the model more realistic and easier to relate to real-life data.

3.1 Standard Conformal Prediction

Conformal Prediction is an effective framework for generating valid prediction intervals that maintain a crucial statistical property called validity coverage. This approach is based on the principle of exchangeability, which means that the sequence of data remains consistent, regardless of how it is arranged. Specifically, given a dataset $(V_1, V_2, ..V_n)$, exchangeability implies that for any permutation π, $(V_1, V_2, ..V_n) =^d (V_{\pi_1}, V_{\pi 2}, ..V_{\pi n})$. In other words, the joint distribution of the data has to be independent of the ordering of the observations; that is why exchangeability is crucial to guarantee the validity of conformal prediction intervals.

The basic procedure, known as Split Conformal Prediction, involves an exchangeable training and calibration dataset; once this is verified, you can proceed with the implementation by first estimating the upper and lower quantiles. An example of Split Conformal Prediction can be given from the Quantile Conformal where the queantiles are computed as, $\hat{q}(X_t; \frac{\alpha}{2})$ and $\hat{q}(X_t; \frac{1-\alpha}{2})$. Once the quantiles are computed, the conformity score on the calibration set can be defined as:

$$S(X_t, y) = \mathrm{m}ax(\hat{q}(X_t; \tfrac{\alpha}{2}) - y; y - \hat{q}(X_t; \tfrac{1-\alpha}{2}))$$

This step finalizes the construction of the prediction interval, satisfying the desired validity guarantees. Consequently, the Critical quantile computation will be given by:

$$\hat{Q}(p) := \inf \left\{ s : \left(\tfrac{1}{|\mathcal{D}_{\text{cal}}|} \textstyle\sum_{(X_r, Y_r) \in \mathcal{D}_{\text{cal}}} \mathbf{1}\{S(X_r, Y_r) \leq s\right) \geq p \right\}$$

The critical quantile is established as the minimum score s such that the proportion of conformity scores within the calibration dataset D_{cal} that are less than or equal to s reaches the desired p-th quantile. This definition serves to establish a score threshold, which ensures that the resulting prediction intervals correspond with the predetermined coverage level. Equivalently, the critical quantile is:

$$S_{\lfloor(1-\alpha)(n_c+1)\rfloor}$$

Where $n_{cal} = |D_{cal}|$ is the size of the calibration dataset, and this is the score at the $(1-\alpha)$ -th quantile in D_{cal}. An alternative formulation of the prediction interval's validity is given by:

$$P(Y_t \in \hat{C}_t) = P(S(X_t, Y_t) \leq \hat{Q}(1-\alpha)) = \frac{\lfloor|\mathcal{D}_{\text{cal}}|(1-\alpha)\rfloor}{|\mathcal{D}_{\text{cal}}|+1}$$

The strength of Conformal Prediction resides in its ability to offer finite-sample validity of the coverage guarantee without the necessity for strong assumptions regarding error distributions, aside from exchangeability. Consequently, Conformal Prediction effectively converts any method of point prediction into a robust procedure for interval prediction, while simultaneously preserving simplicity in implementation and exercising rigorous control over the miscoverage risk.

3.2 Adaptive Conformal Prediction

Standard conformal prediction methods rely on the idea that data patterns stay the same over time.However, in real-world applications, the assumption of exchangeability often fails compared to financial applications, especially under the covariate or distribution shifts. To overcome this limitation, as shown by Gibbs and Candes (2021), Adaptive Conformal Prediction (ACI) can be chosen as a solution to these issues. ACI enhances the traditional conformal prediction framework by adapting it for non-exchangeable time-series data. This is achieved through the dynamic updating of both the base model and the miscoverage level, allowing for the accommodation of potential distributional shifts.

ACI ensures validity through an innovative online forecasting method, which involves constructing conformal inference that adapts to changes in the data as they arise. This necessitates a re-estimation process to align with the most recent observations. The simplicity of this approach lies in its requirement to estimate only a single parameter, denoted as α_t, which contributes to its computational efficiency. Additionally, this method offers the flexibility to integrate various machine learning models for point predictions, further enhancing its applicability and utility. In this scheme introduced by Gibbs and Candes (2021), the adaptive conformal prediction method necessitates the estimation of the score function $S_t()$ while simultaneously updating the critical quantile $\hat{Q}_t$. To fully understand this approach, it is essential to assess the miscoverage rate of prediction, defined as:

$$M_t(\alpha) := P(S_t(X_t, Y_t) > \hat{Q}_t(1-\alpha)$$

Where $M_t(\alpha)$ reflects the proportion of scores that surpass the critical quantile and should ideally approximate α. In case this alignment does not occur, a corrected value α_t in the interval (0,1) is introduced, ensuring that $M_t(\alpha)$ approximates the value of α.

The core of this methodology involves monitoring the miscoverage rate, which necessitates the definition of an indicator function for the errors, expressed as follows:

$$err_t := \begin{cases} 1, & \text{if } Y_t \notin \hat{C}_t(\alpha_t) \\ 0, & \text{otherwise.} \end{cases}$$

Here, the prediction interval is formulated as:

$$\hat{C}_t(\alpha_t) := \{y : S_t(X_t, y) \leq \hat{Q}_t(1-\alpha_t)\}.$$

If the errors are equal to 1, this indicates that the confidence interval is too narrow and fails to encompass all errors. Conversely, if the error equals 0, it suggests that the confidence interval is overly broad and unable to effectively exclude the errors represented by α. If the confidence interval is determined to be too short, the need for adaptation diminishes. In case the err_t equals 0, it signals that the interval is too constricted, necessitating an increase to accommodate more errors. In contrast, if the value is 1, it indicates that the interval is excessively lengthy and requires shortening. A more comprehensive version of this approach incorporates weighted historical errors:

$$\alpha_{t+1} = \alpha_t + \gamma\left(\alpha - \textstyle\sum_{s=1}^{t} w_s \cdot \mathrm{err}_s\right)$$

Where the sequence $w_{s(1\leq s\leq t)}$ comprises weights adding up to 1, thereby prioritizing more recent errors.

Capitolo 4

Use Of CP In Long-memory Processes

This chapter looks into how conformal prediction methods can be applied to time series data that show long-memory behavior. For this application, two different models have been chosen: the $FractionalARIMA$ model and the $StandardARIMA$ model. Applying CP to these two, it can be concluded that when long-range dependence is present, it changes the statistical properties of forecasting. This impacts both the actual predictions we make and the way we create reliable prediction intervals (Beran, 2017). Traditional methods that rely on normality assumptions can struggle with long-memory processes, especially when dealing with finite samples, since the slow convergence of these processes can cause problems.

4.1 Fractional ARIMA Model Specification

$FractionalARIMA$ (Auto-regressive Integrated Moving Average) models can be defined as an extension of the well-known ARIMA framework, allowing processes to exhibit long-range dependence. First,it needs to be recalled the $ARMA$ process definition, an $ARMA(p, q)$ process X_t is mathematically represented by:

$\Phi(B)X_t = \Theta(B)\epsilon_t$

Where B denotes the backshift operator defined by $B(X_t) = X_{t-1}$.
The polynomials $\Phi(B)$, $\Theta(B)$ are defined as:

$\Phi(B) = 1 - \phi_p B^p \ \Theta(B) = 1 + \theta_1 B + ... + \theta_q B^q$

Additionally, the sequence ϵ_t is characterized as white noise with zero mean and variance σ^2. In a situation where the d-th difference is represented as $(1 - B)^d X_t$, adhering to an $ARMA(p,q)$ structure, we refer to the resulting process as an $ARIMA(p,d,q)$:

$\Phi(B)(1-B)^d X_t = \Theta(B)\epsilon_t$

Here d typically holds the value of a non-negative integer, which corresponds to a finite difference that converts a non-stationary time series into a stationary one. Beran (2017) allows d itself to be fractional, which leads to the $FractionalARIMA$ or $FARIMA(p,d,q)$ framework. In this scenario, the relationship is expressed as:

$\Phi(B)(1-B)^d X_t = \Theta(B)\,\varepsilon_t$

But it is interpreted $((1-B)^d$ using its binomial expansion:

$(1-B)^d = \sum_{k=0}^{\infty} \binom{d}{k}(-B)^k$

This formulation represents an infinite-length filter with weights that decay hyperbolically instead of exponentially. The coefficients from this filter decay following a hyperbolic pattern, which contributes to the characteristic long-range dependence observed in these models. When d is equal to or greater than 1, the process is no longer stationary in the conventional sense, and the spectral density near zero becomes non-integrable. Therefore, integer differencing must be applied to the series until d falls back into the stationary range. For instance, if $d = 1.3$, taking the first difference $X_t - X_{t-1}$ results in a fractional ARIMA with $d = 0.3$, then restoring stationarity. An alternative perspective is to view fractional ARIMA processes as either an $ARMA(p,q)$ series that is filtered through the inverse fractional operator.

$(1-B)^d X_t = k_t$

where k_t it is itself an ARMA (p,q) process, or as a purely fractional ARIMA $(0,d,0)$ “innovation” $X'_t = (1-B)^{-d}\varepsilon_t$ processed through an ARMA filter. This duality highlights that long memory can be modeled by either pre-filtering the innovations with fractional methods or post-filtering an ARMA signal. In the frequency domain, the spectral density of X_t takes on the ARMA spectrum $f_{\text{ARMA}}(\omega)$ and is adjusted by a factor of $|1-e^{-i\omega}|^{-2d}$. As ω approaches 0, we get:

$$f(\omega) \;\propto\; |\omega|^{-2d},$$

indicating that for $d > 0$, a pole is present at the origin. This suggests persistent correlations that diminish according to a power law. The practical interest in modeling long-memory processes lies in the range $0 < d < 1$, where the process can remain stationary (for $d < \frac{1}{2}$) or show only mild non-stationarity (for $\frac{1}{2} \leq d < 1$). In this range, ARIMA’s traditional inference and forecasting methods can be applied with minimal modification. This fractional extension was initially proposed independently by Granger and Joyeux in 1980 and by Hosking in 1981, and it has since become fundamental for modeling and estimating long-range dependencies in contemporary time series analysis. His formulation of the ARFIMA(p,d,q) class shows that allowing the differencing order d to take on non-integer values is the same as an aggregation of an infinite-order linear filter. This is precisely the same filter that arises from aggregating an infinite number of short-memory AR(1) components.

4.2 ARIMA Model Specification

ARIMA processes, which stand for Autoregressive Integrated Moving Average processes, are a type of statistical model used for analyzing both stationary and non-stationary data series. This flexibility allows these models to be adapted to various phenomena. Specifically, an ARIMA process consists of three parts: the AR part, which represents autoregressive components; the I component, which applies integration through differentiation; and the MA part, which encompasses the moving average components, which encompasses the moving average components.

A time series Y_t can be defined as an $ARIMA(p, d, q)$ process if it can be differentiated "d" times and it becomes an $ARMA(p, q)$ process:

$$\Delta^d Y_t = \phi_1 \Delta^d Y_{t-1} + ... + \phi_p \Delta^d Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_p \epsilon_{t-q}$$

In this context:

- p refers to the autoregressive order, indicating the number of lagged values of the dependent variable included in the model.

- d denotes the order of differentiation required to make the time series stationary.

- q represents the moving average order, which indicates the number of lagged error terms included in the model.

Differentiation is the process used to remove deterministic or stochastic trends from a time series. The first difference can be defined as:

$$\Delta Y_t = Y_t - Y_{t-1}$$

The difference of order k instead can be obtained recursively:

$$\Delta^k Y_t = \Delta(\Delta^{k-1} Y_t)$$

Another formulation can be obtained using the back-shift operator B, defined as $BY_t = Y_{t-1}$, from which is obtained:

$$\Delta^k Y_t = (1 - B)^k Y_t = \textstyle\sum_{j=0}^{k} \binom{k}{j} (-1)^j Y_{t-j}$$

The order of integration, denoted as d, is a key factor that influences the memory characteristics of ARIMA models. Stationary $ARIMA(p, 0, q)$ processes display mean-reverting tendencies alongside short memory properties. In these models, disturbances gradually dissipate over time, ultimately guiding the series back to its long-term mean. In contrast, integrated processes show long memory traits that depend systematically on the order of integration. Specifically, $ARIMA(p, 1, q)$ processes—commonly used to model log prices, where log returns follow ARMA structures—exhibit persistent memory. In this case, shocks have long-lasting effects on the level of the series, even though the first differences remain stationary. Processes with higher orders of integration, such as $ARIMA(p, 2, q)$, demonstrate even stronger memory properties, characterized by momentum effects that keep the series moving in a particular direction once a trend is established. This hierarchical framework of memory is evident in the distinct behaviors of $I(0)$, $I(1)$, and $I(2)$ processes: $I(0)$ processes tend to revert to their mean, $I(1)$ processes behave similarly to random walks without a fixed reversion point, and $I(2)$ processes exhibit trending behavior with sustained directional movement. Therefore, the degree of integration becomes crucial for understanding how long the effects of innovations persist within the system. This underscores the importance of accurately identifying the parameter d for comprehensive insight and forecasting of the long-term dynamics present in economic and financial time series.

The main limitation of ARIMA models is their reliance on finite-order lags, represented by the parameters p and q. These parameters are used to determine how past values relate to the current value. In Tsay (2005), the author discusses these limitations. For example, it has been demonstrated that autoregressive (AR) processes often exhibit autocorrelation functions (ACFs) that decline exponentially, indicating that the influence of earlier observations diminishes rapidly. This characteristic makes ARIMA ideal for short-term memory processes; however, it limits its ability to handle long-memory situations, where past values—even those from a long time ago—have a significant impact on current outcomes. This is particularly important in fields like financial volatility and macroeconomic trends. Additionally, ARIMA models use the differencing order to achieve stationarity; Tsay (2005) shows that this requirement does not fit well with the fractional dynamics present in long-memory processes.

As discussed in Sections 2.7 and 2.11, true long-range dependence often requires fractional differencing, which ARIMA models cannot directly accommodate. While traditional differencing removes stochastic trends, it overlooks the more complex, slowly decaying autocorrelations that are typical of long-memory characteristics.

Another limitation of ARIMA models is that they require pre-specified lag orders

(p, d, q), which can introduce subjectivity into the modeling process. When the model attempts to capture longer lags, there is a significant risk of overfitting, especially if the values of p and q are increased. That is why Tsay (2005) states that even with adjustments, the finite nature of these models cannot truly replicate the infinite-lag dependencies that are seen in actual long-memory processes. For instance Tsay (2005) has noted that the ARMA, which form the stationary basis for ARIMA models, are built on specific weights that decline geometrically, in contrast with the hyperbolic decay typical of long-memory systems.

4.3 Empirical Evaluation

4.3.1 Simulation Study: ARFIMA Model

A simulation study is essential when exploring long-memory time series models like ARFIMA, as it allows for conducting controlled experiments using known parameters. This simulation approach allows researchers to asses the effectiveness of different statistical methods, validate theoretical properties and understand how estimators behave within finite samples. Different specialized packages can be used in the empirical environment to facilitate the simulation and analysis of fractionally integrated processes.

The principal empirical tools used for ARFIMA modeling are the "fracdiff" and "arfima" packages, each possessing characteristics suited for long-memory analysis. The "fracdiff" package supplies functions specifically for the simulation of fractionally differentiated ARIMA(p,d,q) series, using the function "fracdiff.sim()". This package incorporates the algorithm for fractional differencing and provides robust parameter estimation tools via the Whittle likelihood estimation. Instead, the "arfima" package extends these great features to help with complex modeling tasks. It is primarily used to compute maximum likelihood estimates, forecast several steps, and diagnose models effectively.

```
‘‘‘{r}
library("fracdiff")
library("ggplot2")
library("gridExtra")

set.seed(123)


phi=0.5
theta=0.4
myT=1000
```

```r
d_values=c(0, 0.1, 0.3)



for (myD in d_values) {


  sim_arfima=fracdiff.sim(n=myT, ar=phi, ma=theta, d=myD)
  ts_arfima=sim_arfima$series


  plot_ts_arfima = ggplot(data.frame(time = 1:myT,
                                   value =
                                     ts_arfima),
                          aes(x = time, y = value)) +
    geom_line(color = "steelblue") +
    labs(title =paste( "ARFIMA (1,", myD, ",1) Time Series"),
       x = "Time",
       y = "Value") +
    theme_minimal() +
    theme(plot.title = element_text(size = 12))

  print(plot_ts_arfima)


   acf_result =acf(ts_arfima, lag.max = 50, plot = FALSE)

   acf_df= data.frame(
     lag = 1:50,
     acf = as.numeric(acf_result$acf[2:51])
  )

   conf_limit= 1.96/sqrt(myT)


  acf_plot_arfima = ggplot(acf_df, aes(x = lag, y = acf)) +
    geom_hline(yintercept = 0, color = "black") +
    geom_hline(yintercept = conf_limit, linetype = "dashed",
       color = "red") +
    geom_hline(yintercept = -conf_limit, linetype = "dashed",
       color = "red") +
    geom_col(fill = "darkgreen", color = "darkgreen", alpha =
       0.7, width = 0.8) +
    geom_point(color = "darkgreen", size = 1.5) +
    labs(title = paste("ARFIMA (1,", myD, ",1) Autocorrelation
        Function"),
         x = "Lag",
         y = "ACF") +
    theme_minimal() +
    theme(plot.title = element_text(size = 12)) +
    scale_x_continuous(breaks = seq(0, 50, 5))

  print(acf_plot_arfima)


  spec_arfima = spectrum(ts_arfima, plot = FALSE)

  spec_plot_arfima = ggplot(data.frame(freq =
                                    spec_arfima$freq,spec =
                                      spec_arfima$spec),
                    aes(x = freq, y = spec)) +
  geom_line(color = "red") +
  labs(title = paste("ARFIMA (1,", myD, ",1) Spectral Density"
    ),
       x = "Frequence",
       y = "Spectral Density") +
  theme_minimal()


  grid.arrange(plot_ts_arfima, acf_plot_arfima,
               spec_plot_arfima, nrow = 3)

}

‘‘‘
```

Listing 4.1. ARFIMA Simulation

In this initial empirical study, the prediction of the ARFIMA model is made with the use of alternative values of the fractional differencing parameter. The big section of this code constructs the ARFIMA time series. When we have the simulated series, we are going to study the autocorrelation function (ACF) and spectral densities with the consideration of the parameter d, which influences the amount of memory of the series regarding time. Before entering into the simulation, we assume a few of its significant parameters, which define the significant characteristics of the simulated process. The autoregressive parameter is the parameter employed, and it is set to

0.5. It is a parameter that controls how much current values are influenced by the past ones. We then set the parameter of the moving average, theta, to 0.4, as this determines the magnitude of the past error terms. The simulation, such as analysis, is done on a simulated time series, which comprises $1,000$ observations, which are represented by the parameter myT. An important feature of this simulation is the fractional differencing parameter, d. We take three cases of value 0, 0.1, and 0.5, which denote no memory, short memory, and long memory, respectively. Because it is possible to change the value of d in the ARFIMA process simulation, we will be able to consider how the alterations of this parameter of the model affect the properties of the ARFIMA series. The rest of the script is a loop that simulates various values of d. At every iteration, an ARFIMA(1, d, 1) process is simulated with the 'fracdiff.sim' functional, and the simulated results are shown in a plot.

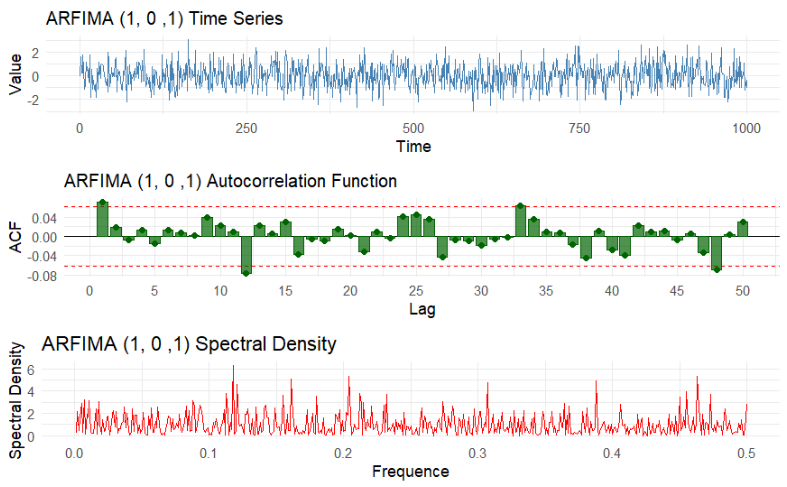

Figura 4.1. Simulated ARFIMA(1,0,1): Time series, ACF, Spectral density

Figure 4.1 represents an $ARFIMA(1,0,1)$ simulation, showing how this particular process behaves in time and which statistical features it owns.

Looking a the time series in the first panel, we can notice how values oscillate randomly around zero for all the 1000 observations. What stands out is the fluctuations' nature, because they do not seem to be completely random; they show some persistence, which suggests the presence of an underlying structure. Values tend to stay in the same region for extended periods before moving to other areas, creating what in statistics is called clustering, so a tendency for similar values to cluster together. In the autocorrelation function shown in the middle panel, we can see how there is a slow decay after the first lag. The first few correlations are positive, indicating that recent values still have some influence on feature values, even after a significant time, which is a typical feature of long memory. The fluctuations between positive and negative values in the following lags also suggest the presence of cyclical components in the behavior of the series. In the third panel spectral density is represented. The concentration in the low frequencies, evidenced by the peaks near zero, confirms that this process is characterized by slow fluctuations. We can see the multiple peaks distributed across the spectrum, which indicate the presence of different periodic components, contributing to the series' behavior. Together, these three graphs give us a complete vision of how long memory shows up, characterized by the time domain through a visible persistence, by the correlation behavior through slow decay, and the domain of low-frequency components.

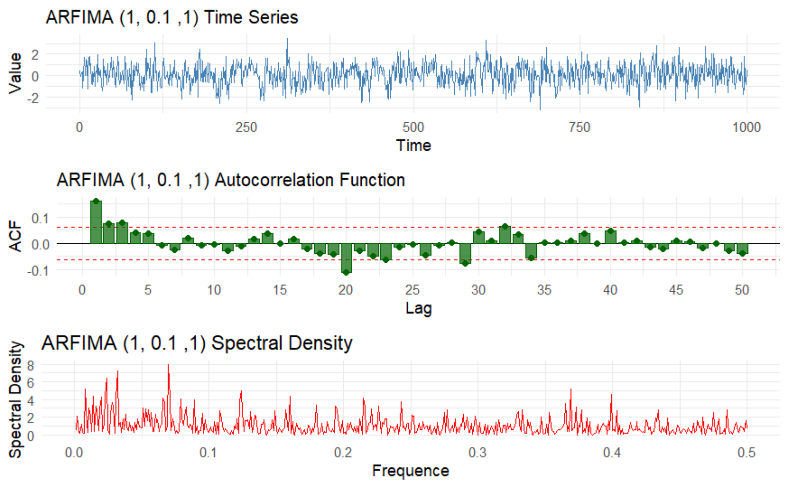

Figura 4.2. Simulated ARFIMA(1,0.1,1): Time series, ACF, Spectral density

The second results representation in Figure 4.2 shows the analysis of a simulated $ARFIMA(1, 0.1, 1)$ time series, where the main difference with the previous case is the value given to the fractional differencing parameter d, used to introduce long memory properties in the process. Looking at the time series in the upward panel, it can be noticed some slight but significant differences concerning the previous case. The values oscillate between

$$-2$$

and +2, but we have better fluctuations because when the time series goes up or down, it tends to stay in the same zone for long periods before switching direction, which proves that the process has more memory than its recent past. The autocorrelation function graph in the center proves that autocorrelation stays positive for more periods than a normal time series, which suggests that events happening today will keep influencing what is about to happen in the future, the main assumption of long memory processes. The spectral density plot shows us a high peak around zero, indicating that most of the process variability comes from slow but persistent movement. The main difference from the previous case is that with a fractional differencing parameter's value of 0.1, we have introduced a visible long memory, while the previous process had developed only the memory typical of standard models; now the process can remember past events for longer lags. This behavior is typical of real phenomena; for example, financial markets often show this kind of persistence, where periods of high volatility are followed by similar periods; natural phenomena such as climatic variations can show long memory, as mentioned in the previous chapters.

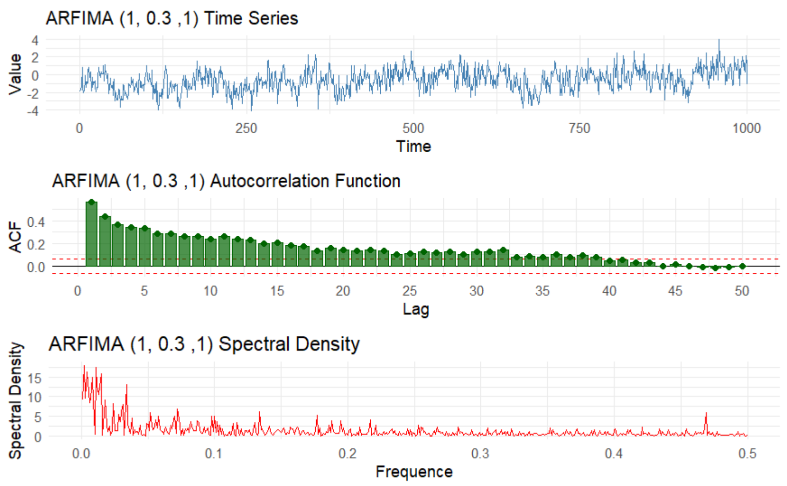

Figura 4.3. Simulated ARFIMA(1,0.3,1): Time series, ACF, Spectral density

Figure 4.3 displays the $ARFIMA(1, 0.3, 1)$ time series where di fractional differencing parameter is set to 0.3, a higher value compared to the previous cases; this change has some visible effect on the series' behavior. Looking at the time series plot,some significative changes can be seen; values oscillate in a bigger range, going to $+4$ an -4, also showing a more persistence behavior since that when it goes up it continues to stay high for long periods, and when it goes down, it stays low for a long time. The autocorrelation function in the middle shows high persistence; the correlations starting high at values over 0.4 fall extremely gradually; even after 30-40 time periods, the correlation is still positive and significant. This means that what is going to happen today will keep influencing future events, even in a month or more, showing an extremely long memory.

The spectral density at the bottom is the most impressive aspect of this graph; we can notice that in the peak near the zero, the frequency is great, reaching values above 15, while all the rest frequencies are flat and close to zero. These results prove that almost the whole energy of the process is concentrated in extremely slow movements. With a value of $d = 0.3$, we are very close to the stationarity limit, which can be found when the fractional differencing parameter is set to 0.5. This process has developed characteristics that make it almost like a random walk, but it still maintains the property of moving back to its mean over the long run; this kind of behavior can be used for modeling phenomena that show very slow, persistent changes.

Comparing these three different cases of an ARFIMA time series simulation, it can be seen how even a single parameter can completely change the time series' behavior. The fractional differencing parameter handles the long memory of the process: with $d = 0$, we obtain a standard process with short memory, setting $d = 0.1$, long memory is introduced, so the time series shows more persistence and past events start influencing the future for longer periods. Lastly, with $d = 0.3$, an extremely long memory is obtained.

4.3.2 Conformal Prediction Simulation

The following empirical study implements the conformal prediction analysis applied to a time series simulated through an ARFIMA (Autoregressive Fractionally Integrated Moving Average) model with different long memory parameters, following the Split Conformal Prediction theoretical approach discussed in Chapter 3. As already explained in the theoretical analysis, the split CP is an appropriate choice for long memory process applications, since it does not assume any specific distributions on the errors. This type of conformal prediction is based on the principle of exchangeability of data and the construction of different prediction intervals through the residuals calibration.

In the ARFIMA models contest, this approach is preferable to traditional methods based on the assumption of having specific distributions. This empirical implementation is based on the Split Conformal Prediction application, which is done by first dividing the dataset into three different subsets: training set, calibration set, and test set. To be more specific, the training set is used for the model estimation, the calibration set to determine the conformal quantiles used in the prediction, and the last set is used to evaluate the model performance. The partition proposed in the empirical code assigns respectively 40%, 40%, and 20% proportions. This approach guarantees a robust calibration and leads us to a reliable statistical valuation, considering that the test set is pretty wide. The split CP analysis is conducted, assigning five different values to the fractional differencing parameter $d(0, 0.1, 0.2, 0.3, 0.4)$, allowing for exploration of how long memory affects the prediction's quality.

The ARFIMA process parameters values have been chosen to represent a controlled case where the autoregressive coefficient $\phi = 0.5$ can introduce a moderate dependence on past values, while the moving average parameter $\theta = 0.4$ can be used to capture past values. In this application, the number of observations has been increased due to the fitting of the ARFIMA model, returning us a reliable calibration. Methodology core is based on the calibration phase, where conformity scores are computed as prediction residuals's absolute values. This approach is based, as seen before in the theoretical analysis, on the score function $|y-\hat{y}|$, which guarantees symmetric intervals around the prediction. After computing the conformity scores, the critical quantile is determined empirically from the calibration set using $(1-\alpha)$ where α is set to 0.2, guaranteeing a confidence level of about 80%. This quantile value represents the threshold over which just a fraction α of the calibration residuals can be considered; this step represents the base for the prediction interval's construction. Performance evaluation is based on two fundamental metrics: the empirical coverage, which measures how many observations fall into the prediction intervals, and the average interval width used to quantify the level of predictive uncertainty. The trade-off between these two metrics is a main aspect in the conformal prediction's analysis for different long memory coefficients. This empirical implementation allows to investigate how time persistence, quantified by parameter d, can influence the prediction's quality. A common result in these cases is that, while implementing the parameter's d value, the interval's width varies in a non-trivial way, that is why we are going to obtain on one side a higher predictability associated to long memory, and on the other side shock persistence could grow the prediction result's variability. The split conformal approach is particularly suitable for this kind of analysis since it provides more validity, which is crucial when working with long memory processes where asymptotic properties may occur slowly. Also, methodology's distribution-free nature makes it more robust when it comes to ARFIMA model's specifications, giving us a reliable framework used to quantify the predictive uncertainty even in the presence of the model's mispecifications.

```{r}
library("fracdiff")
library("forecast")
library("ggplot2")


set.seed(123)
myT = 1000
alpha=0.2
all_results=list()

d_values=c(0,0.1,0.2,0.3,0.4)

for (myD in d_values) {
```

```r

train_size = floor(0.4 * myT)
cal_size = floor(0.4* myT)
test_size = myT - train_size - cal_size


sim = fracdiff::fracdiff.sim(n = myT, ar =0.5, ma = 0.4, d =
    myD)
ts_data = sim$series


train_data = ts_data[1:train_size]
cal_data = ts_data[(train_size + 1):(train_size + cal_size)]
test_data = ts_data[(train_size + cal_size + 1):myT]


train_model=arfima(train_data)


cal_pred=predict(train_model, n.ahead=cal_size)
cal_scores=abs(cal_pred$x-cal_data)


n_c = length(cal_scores)
k = floor((1 - alpha) * (n_c + 1))
critical_quantile = sort(cal_scores)[min(k, n_c)]



test_pred=predict(train_model,n.ahead=test_size)

n_pred=min(length(test_pred$x), test_size)



lower_bound=test_pred$x[1:n_pred] - critical_quantile
upper_bound=test_pred$x[1:n_pred] + critical_quantile


coverage=mean(test_data >= lower_bound & test_data <= upper_
   bound,na.rm=TRUE)

interval_width=mean(upper_bound-lower_bound, na.rm = TRUE)
```

```

  test_scores=abs(test_pred$x-test_data)

  cat("Coverage:",round(coverage,3),"\n")
  cat("Interval Width:",round(interval_width,3),"\n")

  all_results[[paste0("d_",myD)]]=list(
    d=myD,
    coverage=coverage,
    interval_width=interval_width,
    test_data=test_data,
    lower_bound=lower_bound,
    upper_bound=upper_bound,
    test_pred=test_pred$x,
    critical_quantile=critical_quantile
  )


outliers= which(test_data < lower_bound | test_data > upper_
    bound)


plot(test_data, type="l", col="black", lwd=1,
     main=paste("ARFIMA d =", myD, "- Split Conformal"),
     ylab="Value", xlab="Time",
     ylim = range(lower_bound, upper_bound, test_data, na.rm =
         TRUE),
     cex.main = 0.9,
     cex.axis = 0.8,
     cex.lab = 0.8)

lines(test_pred$x, col="red")
lines(lower_bound, col="blue", lty=2)
lines(upper_bound, col="blue", lty=2)


points(outliers, test_data[outliers], pch=16, col="yellow",
    lwd=2, cex=0.5)

legend("topright",
       c("Test data", "Predictions", "Lower bound", "Upper
          bound", "Outliers"),
       col = c("black","red","blue","blue","yellow"),
       lty = c(1,1,2,2,NA),
       pch = c(NA, NA, NA, NA, 16),
       lwd = c(2,1,1,1,2),
       cex = 0.8)
}

```

#Monte Carlo Simulation
library(fracdiff)
library(forecast)
library(parallel)
library(ggplot2)
library(dplyr)
library(tidyr)

set.seed(123)
myT = 1000
alpha = 0.2
d_values = c(0,0.1, 0.2, 0.3, 0.4)
n_rep = 1000

single_splitcp= function(myD) {
  train_size = floor(0.4 * myT)
  cal_size =floor(0.4* myT)
  test_size = myT - train_size - cal_size

  sim = fracdiff::fracdiff.sim(n = myT, ar = 0.5, ma = 0.4, d
     = myD)
  ts_data = sim$series

  train_data = ts_data[1:train_size]
  cal_data = ts_data[(train_size + 1):(train_size + cal_size)]
  test_data = ts_data[(train_size + cal_size + 1):myT]

  model = arfima(train_data)
  cal_pred = predict(model, n.ahead = cal_size)
  cal_scores = abs(cal_pred$x - cal_data)
  n_c = length(cal_scores)
  k = floor((1 - alpha) * (n_c + 1))
  q = sort(cal_scores)[min(k, n_c)]

```
```

```
  test_pred = predict(model, n.ahead = test_size)
  preds = test_pred$x[1:test_size]
  lower =preds - q
  upper = preds + q

  coverage = mean(test_data >= lower & test_data <= upper, na.
     rm = TRUE)
  width = mean(upper - lower, na.rm = TRUE)

  data.frame(d = myD, coverage = coverage, width = width)
}


cl = makeCluster(detectCores() - 1)
clusterExport(cl, varlist = c("myT", "alpha", "n_rep","single_
    splitcp"))
clusterEvalQ(cl, {
  library(fracdiff)
  library(forecast)
  library(dplyr)
})

#100 times for each d
res_list=parLapply(cl, d_values, function(dval) {
  replicate(n_rep, single_splitcp(dval), simplify = FALSE) %>%
    bind_rows()
})
stopCluster(cl)


df = bind_rows(res_list)


summary_per_d = df %>%
  group_by(d) %>%
  summarise(
    mean_coverage = mean(coverage),
    mean_width = mean(width)
  )

print(summary_per_d)


```

```r


df_mean_plot = summary_per_d %>%
  pivot_longer(cols = c(mean_coverage, mean_width),
               names_to = "metric", values_to = "value")

ggplot(df_mean_plot, aes(x = factor(d), y = value, group = 1))
    +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 3) +
  facet_wrap(~ metric, scales = "free_y", ncol = 1,
             labeller = as_labeller(c(mean_coverage = "
                Coverage", mean_width = "Width"))) +
  labs(x = "d Values", y = "Mean", title = "Average Coverage
      and Average Width for each value of d") +
  theme_minimal(base_size = 14)
```

Listing 4.2. Conformal Prediction Simulation

The following results obtained from the application show the Conformal Prediction methodology's performance as the parameter d values change, which will allow us to understand the trade-off between probability coverage and the prediction interval's width. The results represents a behavior in line with the Conformal Prediction theory from chapter 3: the coverage values are between 0.793 with $d = 0.3$ and 0.787 with $d = 0.4$, which are close to the expected confidence interval theoretical value of 0.8. On the other hand, we can notice an upward progression of the interval's width, going from a value of 3.635 ($d = 0$) to 5160 ($d = 0.4$), showing that higher values for the fractional differencing parameter lead to more conservative predictions.

The following plots represent the prediction intervals' distributions for every different value of the parameter d. Starting from the case where d is set to zero, this represents the conformal prediction standard setup, with a coverage value of 0.798 and narrower intervals. The visualization shows a homogeneous distribution of the errors throughout the whole dataset.

While increasing the value of the fractional differencing parameter, a progressive widening of the confidence intervals, represented by the blue lines, can be noticed, it is particularly evident in the $d = 0.4$ case, where intervals' variability is at its maximum with a value of 5.160. The third case in Figure 4.7, instead, where the parameter is set to a value of 0.3, presents a lower coverage of 0.793 with respect to the two previous cases. This suggests that there could be some unstable zones or that the model could have a non-linear behavior for this specific parameter's value, indicating that the relationship between the fractional differencing parameter and

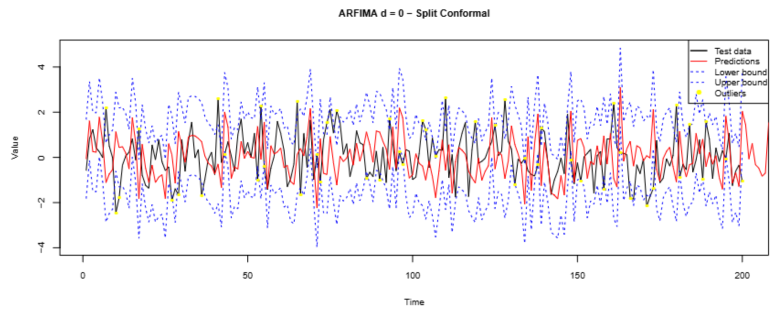

Figura 4.4. Split CP -d=0

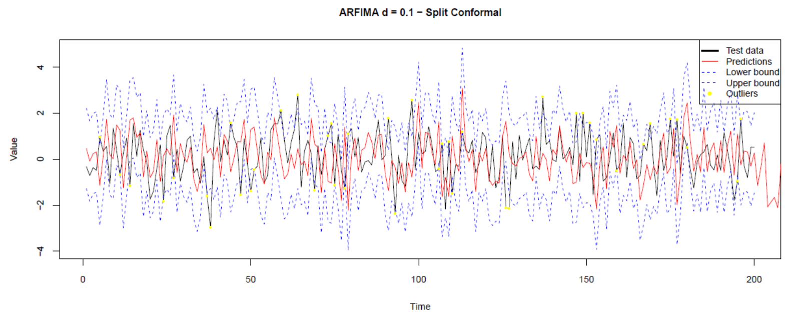

Figura 4.5. Split CP- d=0.1

prediction performance may not be strictly monotonic.

These results suggest to us that the parameter d choice should be projected to the balancing and reliability of the model application. For applications where high precision is required, too low values of d, as $d = 0$ or $d = 0.1$, will return narrower intervals while maintaining a fair coverage.

To asses the performance and stability of Split CP across the different long-memory regimes a Monte Carlo simulation was conducted. The simulation used 1000 observations for each time series, the results , 4.9, show an interesting trade-off between empirical coverage and interval width as the intensity of long-memory

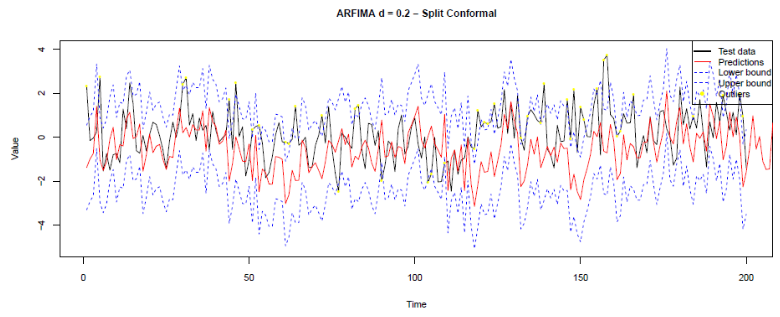

Figura 4.6. Enter Caption

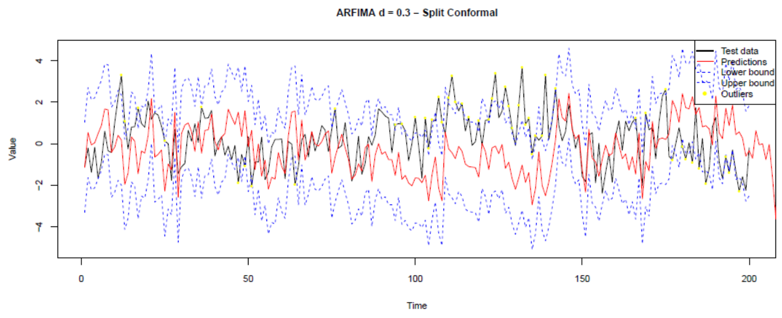

Figura 4.7. Split CP - d=0.3

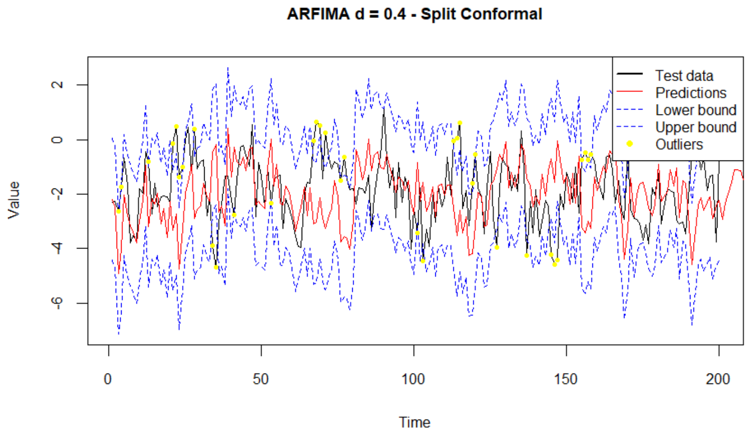

Figura 4.8. Split CP- d=0.4

Parameter d	Empirical Coverage	Interval Width
0.0	0.798	3.635
0.1	0.797	3.730
0.2	0.796	3.955
0.3	0.793	4.369
0.4	0.787	5.160

Tabella 4.1. Empirical Coverage and Interval Width for Different Values of Parameter d

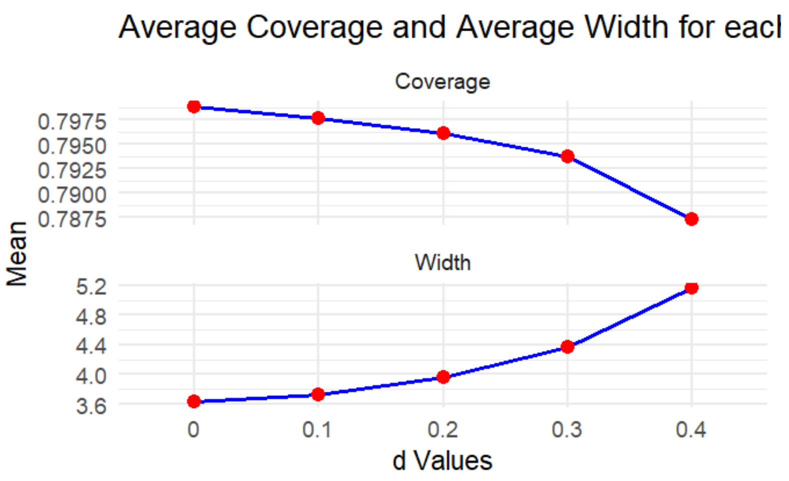

Figura 4.9. Split CP- Avg. Coverage and Avg. Width

changes: in the first plot it can be observed a monotonic decrease of the empirical coverage while the value of d increase, going from 79.8% for $d = 0$ to 78.7% to d=0.4. In spite of this decrease, all the coverage values are close to the nominal value of 80%. The interval width instead increase as the value of d increase, going from 3.6 for $d = 0$ to 5.2 for $d = 0.4$, showing more predictive uncertainty in line with the long memory processes. These results emphasize how Split Conformal Prediction keep the coverage properties stable even in presence of long memory, although a slightly under-coverage level is shown.

Once the implementation of the Conformal Prediction procedure has been carried out, it is essential to proceed with the simulation of the Adaptive Conformal Prediction (ACP) procedure.The code to simulate the Adaptive Conformal Prediction on an artificially generated time series is based on an iterative structure used to compare different configurations of the fractional differencing parameter d, where the main goal is to evaluate the efficacy of conforming predictors when working with a long memory process. In each iteration, an $ARFIMA(p, d, q)$ time series of a fixed length equal to 1000 observations has been generated using the fracdiff. The sim function, as in the previous application, allows inserting a long-term dependency structure for different values of the parameter d, for $d \in [0, 0.4]$.

This simulation includes standard Gaussian errors to keep the framework simple, allowing us to focus on the effects of long-term dependency. In the following step, the simulated time series is split into three different datasets to which a different percentage is assigned for each: 40% training set, 40% calibration set, 20% test set. On the training set in particular, an ARFIMA model is estimated using the function arfima() from the package "arfima", which estimates the model's parameters through the use of the log-likelihood or Whittle likelihood.

The estimated model is then used to generate predictions one step ahead on the calibration set and test set. In the next step, the returns computed between the observed values and the values predicted on the calibration set are used to compute the empirical quantile $q_{1-\alpha}$; this section is core to the ACP methodology, specifically, the empirical quantile formulation is:

$$q_{1-\alpha} = quantile(|\hat{y}_t - y_t|, 1 - \alpha)$$

Where α is the level of significance desired, for example, in our case α is set to a value of 0.2, meaning that the coverage level will be 80%. The computed quantile is then used to build the conformal predictive interval on the test set using the formulation:

$$C_t = [\hat{y}_t - q_{1-\alpha}, \hat{y}_t + q_{1-\alpha}]$$

After computing the intervals, the code proceeds with the computation of the empirical coverage, which represents how many times the observed value falls inside the conformal interval, and the moving average width, indicating the overall uncertainty of the model.

```

library("fracdiff")
library("forecast")
library("ggplot2")
set.seed(123)

phi=0.5
theta=0.4
myT=1000
d_values=c(0, 0.1, 0.2,0.3,0.4)

alpha0=0.2
gamma=0.2

```

```
width_results= data.frame(d = numeric(), mean_width = numeric
    (), median_width = numeric(), sd_width = numeric())

for (myD in d_values) {


  train_size = floor(0.4 * myT)
  cal_size = floor(0.4* myT)
  test_size = myT - train_size - cal_size




  sim = fracdiff::fracdiff.sim(n = myT, ar =0.5, ma = 0.4, d =
      myD)
  ts_data = sim$series


  train_data = ts_data[1:train_size]
  cal_data = ts_data[(train_size + 1):(train_size + cal_size)]
  test_data = ts_data[(train_size + cal_size + 1):myT]


  scores_calib = rep(NA, cal_size)
  for (i in 1:cal_size) {
  data_up_to = c(train_data, cal_data[1:(i-1)])
  model = arfima(data_up_to)
  pred = predict(model, n.ahead=1)$x
  scores_calib[i] = abs(cal_data[i] - pred[1])
}
  scores_sorted = sort(scores_calib, na.last = NA)
  n_c = length(scores_sorted)
  k = floor((1 - alpha0) * (n_c + 1))
  k = min(max(k, 1), n_c)
  q = scores_sorted[k]



  train_model=arfima(train_data)


  all_pred=predict(train_model,n.ahead=cal_size+test_size)$x
  pred_calib=all_pred[1:cal_size]
  pred_test=all_pred[(cal_size+1):(cal_size+test_size)]


  scores_calib=abs(cal_data-pred_calib)


  err=rep(NA,test_size)
  lower_bound=upper_bound=pred_test=rep(NA,test_size)
  alpha_adapt = rep(alpha0, test_size+1)

  for (i in 1:test_size) {
    data_up_to_now = c(train_data, cal_data, test_data[1:(i-1)
       ])
    model = arfima(data_up_to_now)
    pred= predict(model, n.ahead=1)$x
if(length(pred) > 0 && !is.na(pred[1])) {
  pred_test[i] =pred[1]
} else {
  pred_test[i] = NA
}


    scores_sorted = sort(scores_calib, na.last = NA)
    n_c = length(scores_sorted)
    k = floor((1 - alpha_adapt[i]) * (n_c + 1))
    k = min(max(k, 1), n_c)
    q_t = scores_sorted[k]
    lower_bound[i] = pred_test[i] - q_t
    upper_bound[i]=pred_test[i] + q_t



    err[i]=!(test_data[i]>= lower_bound[i]&test_data[i]<=upper
       _bound[i])


    alpha_adapt[i+1]=alpha_adapt[i] + gamma*(alpha0 - err[i])
    alpha_adapt[i+1] = min(max(alpha_adapt[i+1], 0.01), 0.99)

}

interval_width=mean(upper_bound-lower_bound, na.rm = TRUE)

```

```

  cat("Interval Width:",round(interval_width,3),"\n")


  outliers = which(test_data < lower_bound | test_data > upper
     _bound)

  par(mar = c(5, 4, 4, 6), xpd = TRUE)

plot(test_data, type="l", col="black", lwd=1,
     main=paste("ARFIMA d =", myD, "- Adaptive Conformal"),
     ylab="Value", xlab="Time",
     ylim = range(lower_bound, upper_bound, test_data, na.rm =
         TRUE),
     cex.main = 0.9,
     cex.axis = 0.8,
     cex.lab = 0.8)

lines(pred_test, col="red", lwd=1)
lines(lower_bound, col="blue", lty=2)
lines(upper_bound, col="blue", lty=2)

points(outliers, test_data[outliers], pch=16, col="yellow",
    lwd=2, cex=0.5)

legend("topright",
       inset = c(-0.18, 0),
       xpd = TRUE,
       c("Test data", "Predictions", "Lower bound", "Upper
          bound", "Outliers"),
       col = c("black","red","blue","blue","yellow"),
       lty = c(1,1,2,2,NA),
       pch = c(NA, NA, NA, NA, 16),
       lwd = c(2,1,1,1,2),
       cex = 0.8)

cat("\nd =", myD, "| Empirical coverage=", 1-mean(err, na.rm=
    TRUE), "\n")
}
```

Listing 4.3. Conformal Prediction Simulation

The obtained results in terms of empirical coverage and interval width are presented in the following table and graphs for each value of d:

Parameter d	Empirical Coverage	Interval Width
0.0	0.79	2.86
0.1	0.795	3.079
0.2	0.78	3.004
0.3	0.775	3.059
0.4	0.775	2.881

Tabella 4.2. Empirical Coverage and Interval Width for Different Values of Parameter d

From the result 4.2 analysis, we can notice how the coverage performance changes when different values of the fractional differencing parameter are applied. The empirical coverage shows distinctive patterns in the transition from short memory to long memory processes.

In particular, for a value of the parameter d set to zero, which represents a process without long memory, the empirical coverage reaches a value of 79% with an interval width of 2.86. The graph 4.10 shows how the methodology keeps the prediction intervals tighten around the real data ,represented from the black line, and the model's predictions, red line. The outliers displayed with the yellow dots, are distributed in a relatively uniform way through the time series, indicating that the adaptive method is able to capture data variations.

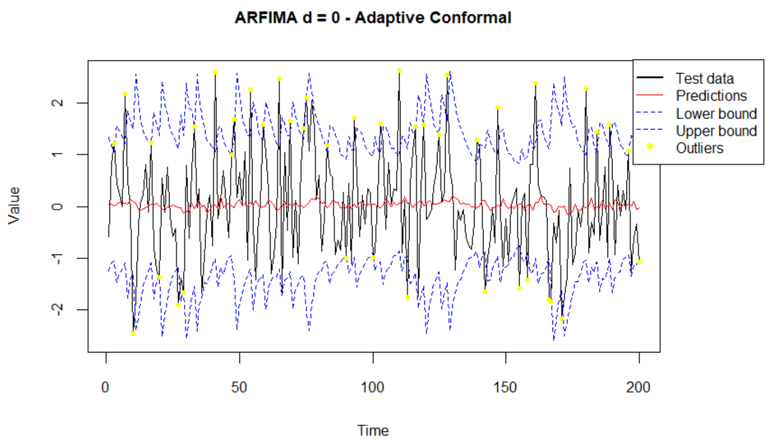

Figura 4.10. ARFIMA Adaptive Conformal - d=0

With $d = 0.1$, a more efficient result for the whole simulation can be observed. The empirical coverage is about 79.5%, together with an increased interval width of 3.079. The graph 4.11 show how the Adaptive CP can take advantage of the slightly persistence in data to optimize the prediction intervals construction. It can be noticed how the interval width is narrowed during periods of more stability while gets widen to maintain the target coverage.

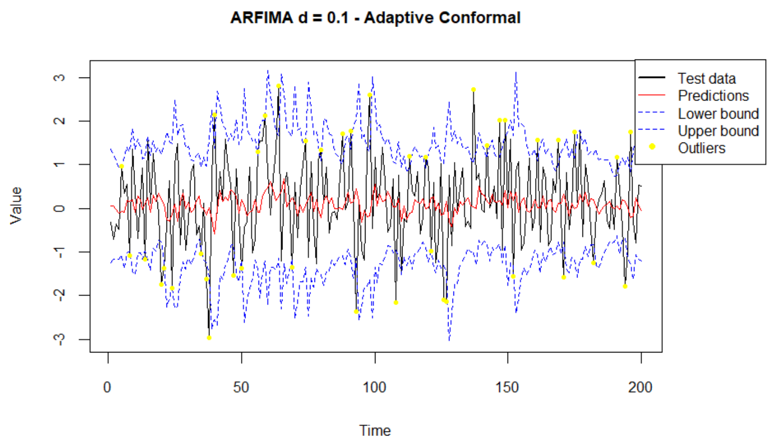

Figura 4.11. ARFIMA Adaptive Conformal - d=0.1

Further increasing the value of d to 0.2, Figure 4.12, the empirical coverage drops slightly to 0.78, which outlines that increasing the model complexity could result into a more variability in the predictions intervals. The blue bands display more wide oscillations. Despite this, the adaptive method is able to follow the data patterns, adapting dynamically the thresholds to handle the process complexity.

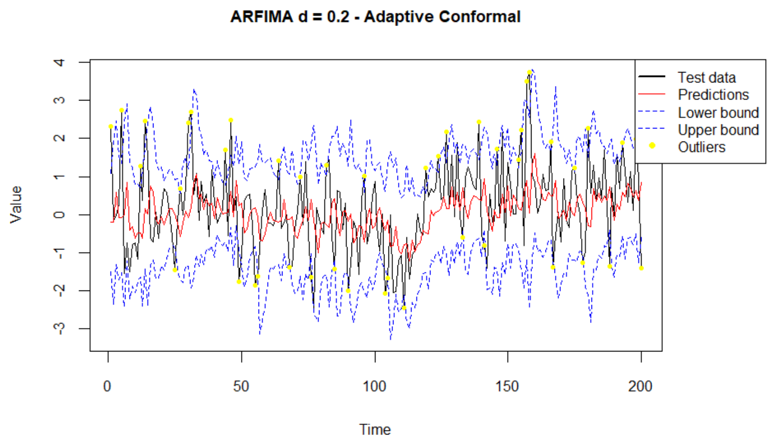

Figura 4.12. ARFIMA Adaptive Conformal - d=0.2

Improving the value of d to 0.3, we can notice a small drop in the empirical coverage to 0.775. The graph 4.13 displays an interesting situation, because despite the process's high persistence the adaptive method is able to control the intervals. The confidence band easily follow the data oscillations, and it can be also noticed how the process identify efficiently periods of more volatility, widening the intervals

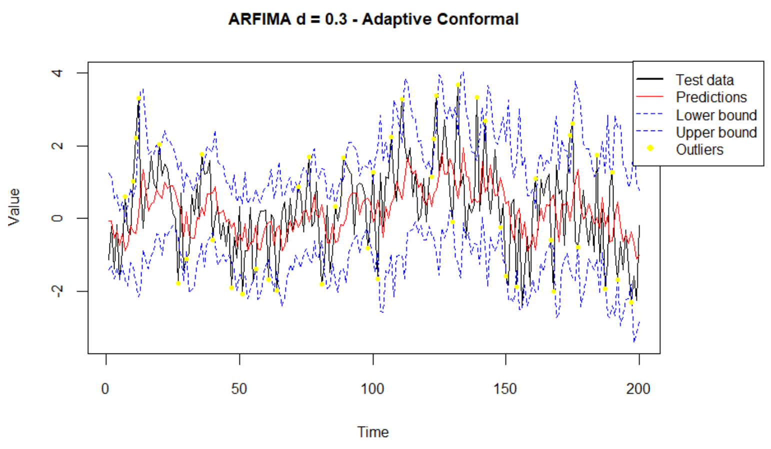

Figura 4.13. ARFIMA Adaptive Conformal - d=0.3

as a consequence. The outliers, represented by the yellow dots, are well distributed, confirming that the method maintains its predictive ability in presence of strong long memory.

The more extreme case is given by the parameter $d = 0.4$, here the empirical coverage remains stable to a value of 77.5% with an interval width of 2.881. The graph shows how the Adaptive Conformal Prediction effectively handles even the process with the highest persistence. The confidence bands shows a significant level of variability

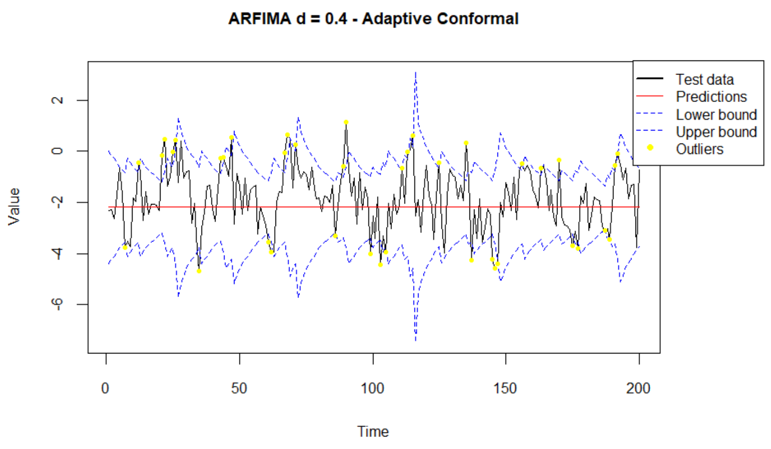

Figura 4.14. ARFIMA Adaptive Conformal - d=0.4

In Figure 4.14, we can see how with a value of d fixed to 0.4, the empirical coverage values increase to 0.805; the graph shows the recovery concerning the value of $d = 0.3$, equalizing the optimal performance obtained for $d = 0$. This pattern followed by the different fractional differencing parameter cases, suggest that while a moderate long memory ($d = 0.2, 0.3$) is challenging for the prediction intervals, with an increased value of d=0.4, we obtain stronger long memory, which provides us with more stable and predictable patterns, that can be catch easily from the model. This relationship between the fractional differencing parameter and the coverage performance highlights the interaction's complexity between long memory characteristics and the effectiveness of conformal prediction in time series forecasting. From the presented graphs, we can also observe the distributions' outliers represented by the yellow dots, confirming the impact of the parameter on the quality of predictions and the robustness of the model.

Looking at the graph ensemble, it is clear why the Adaptive Conformal Prediction Method is preferable to the traditional Split CP. In fact while in the Split CP a progression in the intervals width was observed, going from 3.494 to 4.448, with a variable coverage level between 74.5% and 81.5%; the Adaptive techniques instead displays lower results as seen above.

The model's adaptive capacity is evident from the way the model handles the outliers, since from the graphs it can be seen that the yellow dots are evenly distributed. This feature is fundamental in the financial field where identifying these abnormal behaviors can result into more efficient results. That is why the Adaptive Conformal Prediction can be considered a better technique to quantify uncertainty in a context where the time series are characterized by long-memory.

4.3.3 WTI Crude Oil Simulation

In this paragraph, a real-data application related to the prices of WTI (West Texas Intermediate) oil futures has been conducted to evaluate the efficacy of the Conformal Predictions methods analyzed before.

WTI crude oil prices can be considered a particularly interesting case study due to their long-memory presence and temporal dependency, in line with the results from Shen et al. (2024), which stress the shock persistence and the financial time series autocorrelation, allowing us to use the ARFIMA model and the proposed Conformal Prediction methods.

The use of fractional ARIMA models, as underlined by Shen et al. (2024), helps in understanding the dynamics below WTI oil prices, offering us a statistical framework to look at when observing long-memory effects. As in the previous empirical simulation, the dataset has been divided into: 40% training set, 40% calibration set, and 20% test set; this division will help us keep the dataset's temporal structure, also it is fundamental to ensure that the model's calibration and validation are performed on independent data, maintaining the non-parametric nature typical of the conformal method. The analysis has been conducted using the log-prices to guarantee a more stable variance and also to help improve the time series' statistical properties.

To begin the ARFIMA model estimation, using the "arfima" function from the "fracdiff" package, we estimate an ARFIMA model on the training set. Afterward, the model's predictions are performed on the calibration set at $n.ahead = cal_size$ steps, computing then the conformity scores as the absolute difference between the predicted values and the observed ones, from which we extract the $1 - \alpha$ quantile, representing the threshold used to measure the conformal intervals. The next step is about the precision and evaluation of the test set. The ARFIMA model is applied to predict the test set; in the meantime, using the previously estimated quantile, a symmetric interval is constructed around the prediction. Two key metrics are then calculated, the coverage showing how many test set's observations fall into the computed interval, a good coverage level has a value near to $1 - \alpha$, which in this case is 90%; the second metric to be computed is the average interval width, measuring the prevision's precision.

WTI crude oil application-Split CP

```{r}
library("forecast")
library("ggplot2")
library("fracdiff")

data = read.csv("Crude Oil WTI Futures Historical Data.csv",
    header = TRUE)

head(data)

prices = as.numeric(gsub(",", "", data$Price))
prices = na.omit(prices)
log_prices=log(prices)


set.seed(123)
myT = length(prices)
alpha = 0.1


train_size = floor(0.4 * myT)
cal_size = floor(0.4 * myT)
```

```
test_size = myT - train_size - cal_size


train_data = log_prices[1:train_size]
cal_data = log_prices[(train_size + 1):(train_size + cal_size)
    ]
test_data = log_prices[(train_size + cal_size + 1):myT]

train_model = arfima(train_data)

estimated_d = train_model$d

cat("Estimated d:", round(estimated_d, 4), "\n")


cal_pred = predict(train_model, n.ahead = cal_size)
cal_scores = abs(as.numeric(cal_pred$x) - cal_data)
critical_quantile = quantile(cal_scores, probs = 1 - alpha, na
    .rm = TRUE)


test_pred = predict(train_model, n.ahead = test_size)
pred_values = as.numeric(test_pred$x)[1:length(test_data)]


lower_bound = pred_values - critical_quantile
upper_bound = pred_values + critical_quantile

coverage = mean(test_data >= lower_bound & test_data <= upper_
    bound, na.rm = TRUE)

interval_width =mean(upper_bound - lower_bound, na.rm = TRUE)

cat("Estimated d =",estimated_d, " Coverage =", round(coverage
    ,3), "  Interval width =", round(interval_width,3), "\n")


plot_data = data.frame(
  index = 1:length(test_data),
  test_data = test_data,
  test_pred = pred_values,
  lower_bound = lower_bound,
  upper_bound = upper_bound
)

plot_oil = ggplot(plot_data, aes(x = index)) +
  geom_ribbon(aes(ymin = lower_bound, ymax = upper_bound, fill
      = "Confidence Interval"),
              fill = "skyblue", alpha = 0.4) +
  geom_line(aes(y = test_pred, color = "Prediction"), size =
     1) +
  geom_point(aes(y = test_data, color = "Observed"), size = 1,
      alpha = 0.8) +
  scale_color_manual(values = c("Prediction" = "red", "
     Observed" = "black")) +
  scale_fill_manual(values = c("Confidence Interval" = "
     skyblue")) +
  labs(title = "WTI Crude oil Split CP (Log prices)",
       subtitle = paste("estimated d =", round(estimated_d, 4)
          ,
                        "| Coverage =", round(coverage, 3)),
       x = "Index", y = "Log-Value",
       color = "", fill = "") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))

print(plot_oil)



```

WTI crude oil application-Adaptive CP

```{r}
library("forecast")
library("ggplot2")
library("fracdiff")

data = read.csv("Crude Oil WTI Futures Historical Data.csv",
   header = TRUE)
head(data)
prices = as.numeric(gsub(",", "", data$Price))
prices = na.omit(prices)
```

```
log_prices = log(prices)

set.seed(123)
myT = length(prices)
alpha = 0.1

gamma = 0.01




train_size = floor(0.4 * myT)
cal_size = floor(0.4 * myT)
test_size = myT - train_size - cal_size

train_data = log_prices[1:train_size]
cal_data = log_prices[(train_size + 1):(train_size + cal_size)
    ]
test_data = log_prices[(train_size + cal_size + 1):myT]

cat("=== Parameter d estimation ===\n")
train_model = arfima(train_data)

estimated_d = train_model$d
cat("Estimated parameter:", round(estimated_d, 4), "\n")



cal_pred = predict(train_model, n.ahead = cal_size)
cal_scores = abs(as.numeric(cal_pred$x) - cal_data)


initial_quantile = quantile(cal_scores, probs = 1 - alpha, na.
    rm = TRUE)

adaptive_quantile = initial_quantile
test_pred = predict(train_model, n.ahead = test_size)
pred_values = as.numeric(test_pred$x)[1:length(test_data)]

lower_bounds = numeric(length(test_data))
upper_bounds = numeric(length(test_data))
quantile = numeric(length(test_data))
```

```

for (t in 1:length(test_data)) {
  lower_bounds[t] = pred_values[t] - adaptive_quantile
  upper_bounds[t] = pred_values[t] + adaptive_quantile


  quantile[t] = adaptive_quantile


  new_score = abs(pred_values[t] - test_data[t])

  if (test_data[t] < lower_bounds[t] || test_data[t] > upper_
      bounds[t]) {

    adaptive_quantile = adaptive_quantile + gamma * (new_score
        - adaptive_quantile)
    } else {
      adaptive_quantile = adaptive_quantile - gamma * 0.1 * (
          adaptive_quantile - new_score)
    }
  adaptive_quantile = max(adaptive_quantile, 0.001)
  coverage_values[t] = as.numeric(test_data[t] >= lower_bounds
      [t] & test_data[t] <= upper_bounds[t])
  }


coverage = mean(coverage_values, na.rm = TRUE)
interval_width = mean(upper_bounds - lower_bounds, na.rm =
    TRUE)


cat("=== ADAPTIVE CP RESULTS ===\n")
cat("estimated d value =", round(estimated_d, 4), " Coverage =
    ", round(coverage, 3),
      "  Avg Interval width =", round(interval_width, 3),
      "  Initial quantile =", round(initial_quantile, 3),
      "  Final quantile =", round(adaptive_quantile, 3), "\n\n
          ")

plot_data = data.frame(
  index = 1:length(test_data),
  test_data = test_data,
  test_pred = pred_values,
```

```r
  lower_bound = lower_bounds,
  upper_bound = upper_bounds,
  coverage_value = coverage_values
)

plot_oil = ggplot(plot_data, aes(x = index)) +
  geom_ribbon(
    aes(ymin = lower_bound, ymax = upper_bound, fill = "
       Prediction Interval"),
    alpha = 0.4
  ) +
  geom_line(
    aes(y = test_pred, color = "Prediction"),
    linewidth = 1
  ) +
  geom_point(
    aes(y = test_data, color = "Actual"),
    size = 1, alpha = 0.8
  ) +
  scale_color_manual(
    name = "Legend",
    values = c("Prediction" = "red", "Actual" = "black")
  ) +
  scale_fill_manual(
    name = "Legend",
    values = c("Prediction Interval" = "lightcoral")
  ) +
  labs(
    title = paste("WTI crude oil Adaptive CP- Log prices-
       Estimated d =", round(estimated_d, 4)),
    subtitle = paste("Coverage =", round(coverage, 3),
                     "| Avg Width =", round(interval_width, 3)
                        ),
    x = "Time Index", y = "Log-Price"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5)
  )


print(plot_oil)
```

```

coverage_data = data.frame(index = 1:length(test_data),
    coverage = coverage_values)

coverage_plot = ggplot(coverage_data, aes(x = index, y =
    coverage)) +
  geom_line(color = "blue", linewidth = 1) +
  labs(title = "Coverage Over Time",
       subtitle = paste("Mean Coverage:", round(coverage, 3)),
       x = "Time Index", y = "Coverage") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))

print(coverage_plot)

  width_data = data.frame(
    index = 1:length(test_data),
    width = upper_bounds - lower_bounds
  )

  width_plot = ggplot(width_data, aes(x = index, y = width)) +
    geom_line(color = "green", linewidth = 1) +
    geom_hline(yintercept = mean(width_data$width),
               color = "darkgreen", linetype = "dashed") +
    labs(title = paste("Interval Width Evolution - d stimato =
        ", round(estimated_d, 4)),
         subtitle = paste("Average Width:", round(mean(width_
             data$width), 3)),
         x = "Time Index", y = "Interval Width") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))

print(width_plot)

results = list(
  d_estimated = estimated_d,
  coverage = coverage,
  avg_width = interval_width,
  initial_quantile = initial_quantile,
  final_quantile = adaptive_quantile,
  quantile = quantile,
  model_summary = summary(train_model)
)



cat("\n=== ADAPTIVE CP FINAL SUMMARY ===\n")
cat("Estimated parameter d:", round(estimated_d, 4), "\n")
cat("Coverage:", round(coverage, 3), "\n")
cat("INtervals average width:", round(interval_width, 3), "\n"
    )
cat("Initial threshold:", round(initial_quantile, 3), "\n")
cat("Final threshold:", round(adaptive_quantile, 3), "\n")
cat("Threshold difference:", round(adaptive_quantile - initial
    _quantile, 4), "\n")
‘‘‘
```

Listing 4.4. Conformal Prediction Simulation

Figure 4.15 displays the analysis of the WTI crude oil log-prices' time series, with a focus on the fractional differencing parameter.

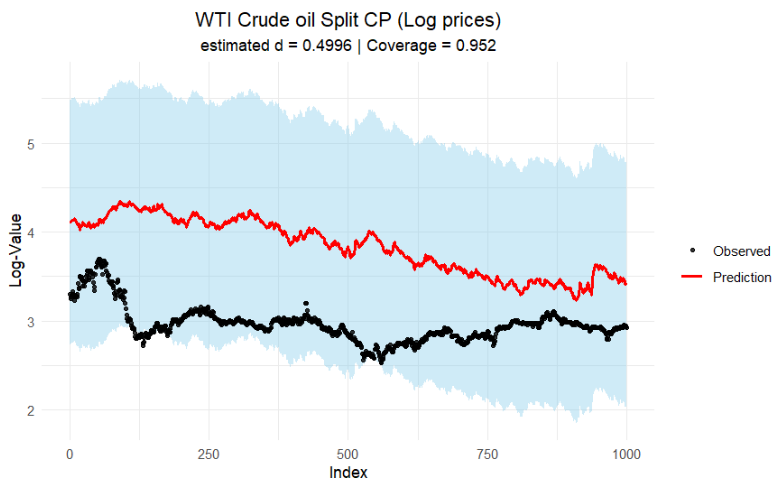

Figura 4.15. WTI Crude Oil-Split CP

The fractional differencing parameter estimated value is 0.4996, a value close to 0.5, indicating that the WTI oil time series exhibits very strong long memory characteristics.

It is also important to understand that in this simulation, we are applying real data, so it is essential for the parameter d to be automatically estimated and not chosen in advance, because time dependence has to be determined empirically. The estimated parameter suggests that the time series is almost on the line between stationarity and non-stationarity. We can also notice how oil price shocks have persistent effects that fade very slowly over time, and finally, the time series displays a time dependence structure typical of financial commodity markets.

The statistical analysis confirms that the obtained results are in line with the ones expected. The high coverage level of about 95.2% indicates that in almost 95 cases out of 100, the computed confidence interval will include the true value assigned to the fractional differencing parameter. The interval width, on the other hand, presents a value of 2.746, which can be considered sufficiently small to ensure a good accuracy in identifying the degree of fractional integration. Looking at the presented graph, we can see how two distinct dynamics emerge from the oil prices. In the first part of the series (indices 0-200), the prices show more volatility with significant oscillations around the log-value of 4, which corresponds to 55 dollars per barrel. In the second half of the chart, we can notice how there is a more gradual downward trend, leading prices to stabilize around the value of 3, corresponding to 20 dollars per barrel. The blue confidence band maintains a constant width for the whole time frame, indicating that process variability stays the same even if we observe some price changes, this behavior is in line with the long memory presence identified in the model.

From a financial point of view, an estimated value of d near 0.5 reveals some special features of the oil markets that have important consequences. Prediction's results indicated to us that while short-term prices have unpredictable movements, there are persistence effects that occur over longer time horizons. This estimation methodology proves to be particularly useful in the analysis of financial data, as it allows the structural characteristics of the time series to be identified objectively without having to resort to arbitrary assumptions.

Figure 4.16 present the results from the implementation of a Adaptive Conformal Prediction algorithm applied on the same time series of the WTI Crude Oil prices, using the automatically estimated, from the ARFIMA model, fractional differencing parameter $d = 0.4996$.

The algorithm implemented shows a strategy of dynamic adaptation of the prediction intervals. The system divides the data into three datasets: training set 40%, calibration 40%, and test 20%, allowing a robust validation of the predictive performance. The adaptation mechanism acts in a specific way: when an observation falls outside the confidence interval, the adaptive quantile is incremented to get a wider prediction band; on the other hand, when the predictions are accurate, the intervals get narrower. From the graph, we can see that the coverage value of 91.8% is slightly lower than the theoretical target value of 95%, but can still be considered a satisfying result given the fact that oil markets are very volatile. The average interval width presents a value of 2.507, representing a good compromise between

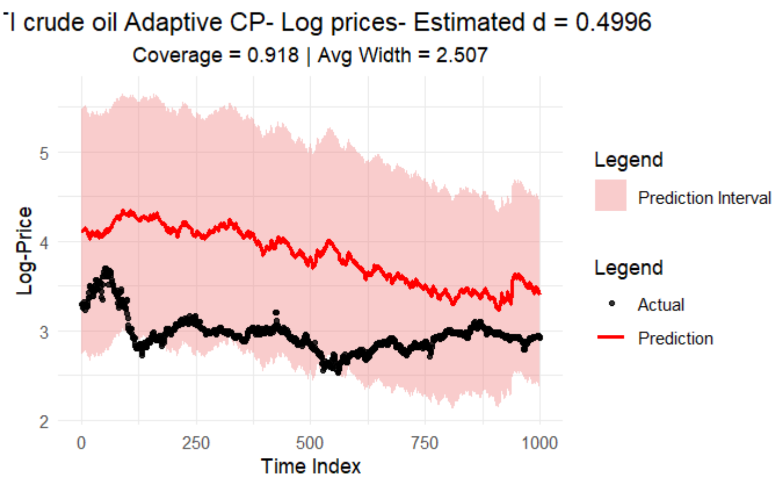

Figura 4.16. WTI Crude Oil- Adaptive CP

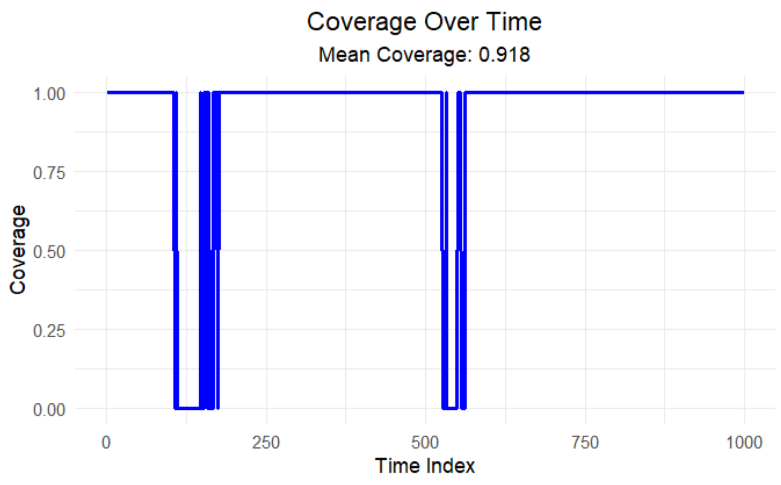

Figura 4.17. WTI Cude Oil- Coverage

accuracy and reliability of predictions.

4.3.4 SP 500 Simulation

This empirical simulation represents a study about the SP 500 Index log-return, based on the analysis of historical data going from 1° January 2000 to the present day. Using the ARFIMA model, the goal is to highlight long memory characteristics present in these financial data and also to make predictions about future returns. This approach is not only relevant from a statistical point of view, but it is also important for financial investors and analysts from a practical point of view.

If we consider the general financial market environment, the SP 500 index is one of the most well-known from a global point of view, and it is also one of the

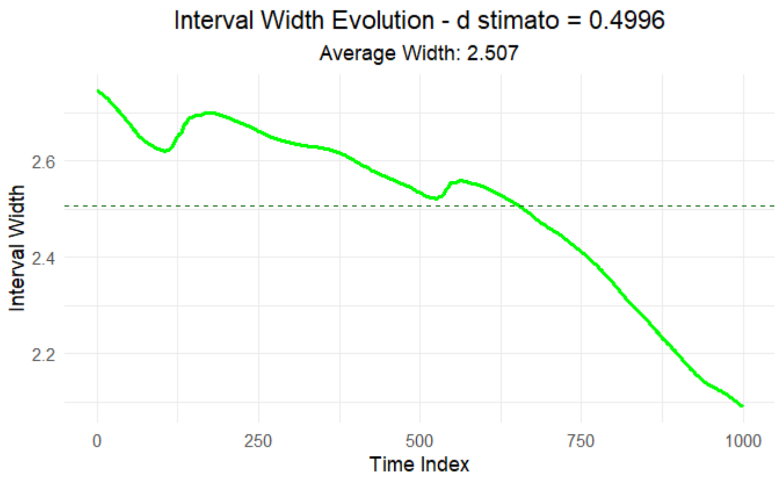

Figura 4.18. WTI Crude Oil - Interval

most important American indices. This index was created by Standard Poor's in 1957, and it tracks the performance of a stock basket formed by the 500 largest-capitalization U.S. companies. Its fluctuations can be influenced by different factors, which can be economic, political, or social, making its analysis fundamental in the financial market.

As we have seen in Chapter 4, the ARFIMA model is particularly efficient in capturing the dynamics of time series affected by long memory, implying that past events or past shocks will keep influencing future returns even after a long time. This characteristic is particularly suitable in the financial markets, where historical events can still have an influence even after years. That is why my goal is to understand how the past can keep influencing the present and also make accurate predictions on future returns using the ARFIMA model. The simulation starts with the upload of the SP 500 index historical data using the function "getSymbols"; once the upload is completed, the prices are converted into logarithmic returns. This transformation is essential because it helps stabilize the variance and make sure that the time series is stationary. After the data preparation, these are divided into the three usual sets: data, calibration, and test; this partition is essential to reduce the overfitting risk. In the financial context, this phase is crucial because markets can exhibit cyclical and nonlinear behavior. Afterward, the ARFIMA model is trained on the training data. In this simulation, the estimated parameter d has a value of 0.0097, which can be considered a significant result. This value indicates that to achieve stationarity the differentiation needed is minimal, suggesting that there are persistent long-term correlations in the returns. Once we have trained the model, it is used to make predictions about future returns; during this process, we also compute the confidence intervals for the predictions, so that we can predict the associated uncertainty.

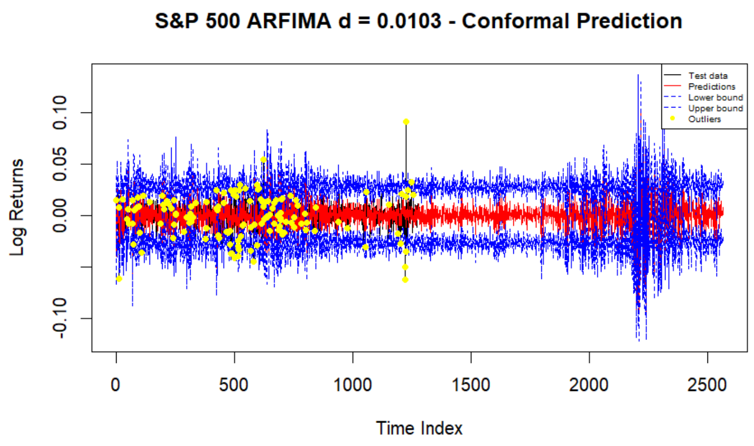

Figura 4.19. SP 500-Split Conformal Prediction

In Figure 4.19, we can see how the outliers are highlighted by the yellow dots; these represent values that deviate significantly from the model predictions. In a financial context, the outliers are used to represent rare events, such as economic crisis, geopolitical news, or even drastic changes in monetary policy, so identifying them is crucial to question predictions' efficiency.

```{r}
library("forecast")
library("ggplot2")
library("fracdiff")
library("quantmod")


set.seed(123)
alpha=0.1

getSymbols("^GSPC", from = "2000-01-01", auto.assign = TRUE)
sp500_prices = na.omit(Cl(GSPC))
sp500_log_prices = log(sp500_prices)
sp500_log_returns = diff(sp500_log_prices)
ts_data = as.numeric(sp500_log_returns)
ts_data = ts_data[!is.na(ts_data)]

myT = length(ts_data)


train_size = floor(0.4 * myT)
```

```
cal_size = floor(0.4 * myT)
test_size = myT - train_size - cal_size


train_data = ts_data[1:train_size]
cal_data = ts_data[(train_size + 1):(train_size + cal_size)]
test_data = ts_data[(train_size + cal_size + 1):myT]

train_model = arfima(train_data)
estimated_d = train_model$d

cat("Estimated d:", round(estimated_d, 4), "\n")

cal_pred = predict(train_model, n.ahead = cal_size)
cal_scores = abs(as.numeric(cal_pred$x) - cal_data)
critical_quantile = quantile(cal_scores, probs = 1 - alpha, na
    .rm = TRUE)

test_pred = predict(train_model, n.ahead = test_size)
pred_values = as.numeric(test_pred$x)[1:length(test_data)]

lower_bound = pred_values - critical_quantile
upper_bound = pred_values + critical_quantile

valid_idx = which(!is.na(test_data) & !is.na(pred_values))

test_data_valid    = test_data[valid_idx]
pred_values_valid  = pred_values[valid_idx]
lower_bound_valid  = lower_bound[valid_idx]
upper_bound_valid  = upper_bound[valid_idx]
test_dates_valid   = dates[(train_size + cal_size + 1):myT][
    valid_idx]

coverage = mean(test_data_valid >= lower_bound_valid & test_
    data_valid <= upper_bound_valid)
interval_width = mean(upper_bound_valid - lower_bound_valid)


outliers = which(test_data_valid < lower_bound_valid | test_
    data_valid > upper_bound_valid)


plot(test_data_valid, type="l", col="black", lwd=1,
     main=paste("S&P 500 ARFIMA d =", round(estimated_d, 4), " - Conformal Prediction"),
     ylab="Log Returns", xlab="Time Index",
     ylim = range(lower_bound_valid, upper_bound_valid, test_data_valid, na.rm = TRUE),
     cex.main = 0.9,
     cex.axis = 0.8,
     cex.lab = 0.8)


lines(pred_values_valid, col="red", lwd=1)


lines(lower_bound_valid, col="blue", lty=2)
lines(upper_bound_valid, col="blue", lty=2)


if(length(outliers) > 0) {
  points(outliers, test_data_valid[outliers], pch=16, col="yellow", cex=0.8)
}


legend("topright",
       c("Test data", "Predictions", "Lower bound", "Upper bound", "Outliers"),
       col = c("black","red","blue","blue","yellow"),
       lty = c(1,1,2,2,NA),
       pch = c(NA, NA, NA, NA, 16),
       lwd = c(1,1,1,1,NA),
       cex = 0.5)


cat("\n=== OUTLIER ANALYSIS ===\n")
cat("Estimated d =", round(estimated_d, 4), "\n")
cat("Empirical coverage =", round(coverage, 4), "\n")
cat("Target coverage =", 1-alpha, "\n")
cat("Number of outliers =", length(outliers), "out of", length(test_data_valid), "\n")
cat("Outlier rate =", round(length(outliers)/length(test_data_valid), 4), "\n")


```

```
if(length(outliers) > 0) {
  cat("\nFirst few outliers:\n")
  n_show = min(10, length(outliers))
  for(i in 1:n_show) {
    idx = outliers[i]
    cat("Index", idx, ": Actual =", round(test_data_valid[idx
       ], 4),
        ", Predicted =", round(pred_values_valid[idx], 4),
        ", Lower =", round(lower_bound_valid[idx], 4),
        ", Upper =", round(upper_bound_valid[idx], 4), "\n")
  }
  if(length(outliers) > 10) {
    cat("... and", length(outliers) - 10, "more outliers\n")
  }
}



```

```{r}
library("forecast")
library("ggplot2")
library("fracdiff")
library("quantmod")

getSymbols("^GSPC", from = "2000-01-01", auto.assign = TRUE)
sp500_prices = na.omit(Cl(GSPC))
sp500_log_prices = log(sp500_prices)
sp500_log_returns = diff(sp500_log_prices)
ts_data = as.numeric(sp500_log_returns)
ts_data = ts_data[!is.na(ts_data)]

set.seed(123)
myT=length(ts_data)
alpha=0.1
gamma=0.01

train_size = floor(0.4 * myT)
cal_size = floor(0.4 * myT)
test_size = myT - train_size - cal_size


```

```
train_data = ts_data[1:train_size]
cal_data = ts_data[(train_size + 1):(train_size + cal_size)]
test_data = ts_data[(train_size + cal_size + 1):myT]

train_model = arfima(train_data)
estimated_d = train_model$d

cat("Estimated d:", round(estimated_d, 4), "\n")

cal_pred =predict(train_model, n.ahead = cal_size)
cal_scores = abs(as.numeric(cal_pred$x) - cal_data)

initial_quantile =quantile(cal_scores, probs = 1 - alpha, na.
    rm = TRUE)
adaptive_quantile = initial_quantile
test_pred = predict(train_model, n.ahead = test_size)
pred_values = as.numeric(test_pred$x)[1:length(test_data)]


lower_bounds = numeric(length(test_data))
upper_bounds = numeric(length(test_data))
quantile_values = numeric(length(test_data))

for (t in 1:length(test_data)) {
  lower_bounds[t] = pred_values[t] - adaptive_quantile
  upper_bounds[t] = pred_values[t] + adaptive_quantile
  quantile_values[t] = adaptive_quantile

  new_score = abs(pred_values[t] - test_data[t])

  if (test_data[t] < lower_bounds[t] || test_data[t] > upper_
      bounds[t]) {
    adaptive_quantile = adaptive_quantile + gamma * (new_score
        - adaptive_quantile)
  } else {
    adaptive_quantile = adaptive_quantile - gamma * 0.1 * (
        adaptive_quantile - new_score)
  }
  adaptive_quantile = max(adaptive_quantile, 0.001)
}


coverage = mean(test_data >= lower_bounds & test_data <= upper
    _bounds , na.rm = TRUE)
interval_width = mean(upper_bounds - lower_bounds , na.rm =
    TRUE)

outliers = which(test_data_valid < lower_bound_valid | test_
    data_valid > upper_bound_valid)

cat("=== S&P500 ADAPTIVE CP RESULTS ===\n")
cat("Estimated parameter d =", round(estimated_d, 4), "
    Coverage =", round(coverage , 3),
     "  Average interval width =", round(interval_width , 6),
     "  Initial quantile =", round(initial_quantile , 6),
     "  Finale quantile =", round(adaptive_quantile , 6), "\n\n"
        )


plot_data = data.frame(
  index = 1:length(test_data),
  test_data = test_data,
  test_pred = pred_values,
  lower_bound = lower_bounds,
  upper_bound = upper_bounds,
  quantile_value = quantile_values
)

plot(test_data_valid, type = "l", col = "black", lwd = 1,
     main = paste("S&P 500 ARFIMA d =", round(estimated_d, 4),
          "- Conformal Prediction"),
     ylab = "Log Returns", xlab = "Time Index",
     ylim = range(lower_bound_valid, upper_bound_valid, test_
         data_valid, na.rm = TRUE),
     cex.main = 0.9,
     cex.axis = 0.8,
     cex.lab = 0.8)

lines(pred_values_valid, col = "red", lwd = 0.5)
lines(lower_bound_valid, col = "blue", lty = 1)
lines(upper_bound_valid, col = "blue", lty = 1)

if (length(outliers) > 0) {
  points(outliers, test_data_valid[outliers], pch = 16, col =
      "yellow", cex = 0.8)
}
```

```r

legend("topright",
       legend = c("Test data", "Predictions", "Lower bound", "
          Upper bound", "Outliers"),
       col = c("black", "red", "blue", "blue", "yellow"),
       lty = c(1, 1, 2, 2, NA),
       pch = c(NA, NA, NA, NA, 16),
       lwd = c(0.5,0.5, 0.5, 0.5, NA),
       cex = 0.5)

coverage_vector = as.numeric(test_data >= lower_bounds & test_
    data <= upper_bounds)
cumulative_coverage = cumsum(coverage_vector) / (1:length(
    coverage_vector))

coverage_data = data.frame(
  index = 1:length(test_data),
  cumulative_coverage = cumulative_coverage
)

coverage_plot = ggplot(coverage_data, aes(x = index, y =
    cumulative_coverage)) +
  geom_line(color = "darkorange", linewidth = 1) +
  geom_hline(yintercept = 1 - alpha, color = "darkred",
      linetype = "dashed", linewidth = 0.8) +
  labs(title = paste("Cumulative Coverage - d stimato =",
      round(estimated_d, 4)),
       subtitle = paste("Target:", 1 - alpha, "| Final:",
          round(tail(cumulative_coverage, 1), 3)),
       x = "Time Index", y = "Cumulative Coverage") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))

print(coverage_plot)

width_data = data.frame(
  index = 1:length(test_data),
  width = upper_bounds - lower_bounds
)
width_plot = ggplot(width_data, aes(x = index, y = width)) +
  geom_line(color = "green", linewidth = 1) +
  geom_hline(yintercept = mean(width_data$width),
              color = "darkgreen", linetype = "dashed") +
  labs(title = paste("Interval Width Evolution - d stimato =",
       round(estimated_d, 4)),
        subtitle = paste("Average Width:", round(mean(width_
           data$width), 6)),
        x = "Time Index", y = "Interval Width") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
         plot.subtitle = element_text(hjust = 0.5))
print(width_plot)

cat("\n=== ADAPTIVE CP FINAL SUMMARY ===\n")
cat("Estimated parameter d:", round(estimated_d, 4), "\n")
cat("Coverage:", round(coverage, 3), "\n")
cat("Intervals average width:", round(interval_width, 6), "\n"
    )
cat("Initial threshold:", round(initial_quantile, 6), "\n")
cat("Final threshold:", round(adaptive_quantile, 6), "\n")
cat("Threshold difference:", round(adaptive_quantile - initial
    _quantile, 6), "\n")
‘‘‘
```

Listing 4.5. Conformal Prediction Simulation

The graph generated from the Split Conformal Prediction analysis (Figure 4.19) gives us a visual representation of the results. On the y-axis we represent the logarithmic returns, while on the x-axis the time. The black lines in the graph represent the real data, while the red lines represent the predictions made using the ARFIMA model, which highlight how the model can capture general trends and can adapt to variations. The upper and lower bound displayed with the two dashed blue lines provides a margin of uncertainty around the forecast. Representing the time series in this way is fundamental for the investors since it helps in the evaluation of the risk associated with the decisions taken based on these provisions. Finally, the outliers are represented with the yellow dots and report the areas that need further analysis.

In conclusion, this simulation study is important since it not only provides us with provisions of the SP 500 future returns, but also gives us a bigger picture of the index's long-term behavior, which is why applying the ARFIMA model to analyze long memory can help improve investment decisions.

The next analysis focuses on the use of the Adaptive Conformal Prediction applied to the logarithmic returns of the SP 500 index, always in a long memory context.

The Adaptive Conformal Prediction is based on a statistical framework whose aim is to provide confidence intervals and predictions that adapt dynamically to the data. This approach is particularly useful in financial markets where the information's uncertainty can significantly influence investment decisions. As in the previous analysis, we start from the index data preparation by converting prices into logarithmic returns; this transformation allows us to have a better interpretation of the market fluctuations. The conformal approach also requires data stratification similar to that seen with the Split Conformal Prediction. Figure 4.20 displays the Cumulative Coverage.

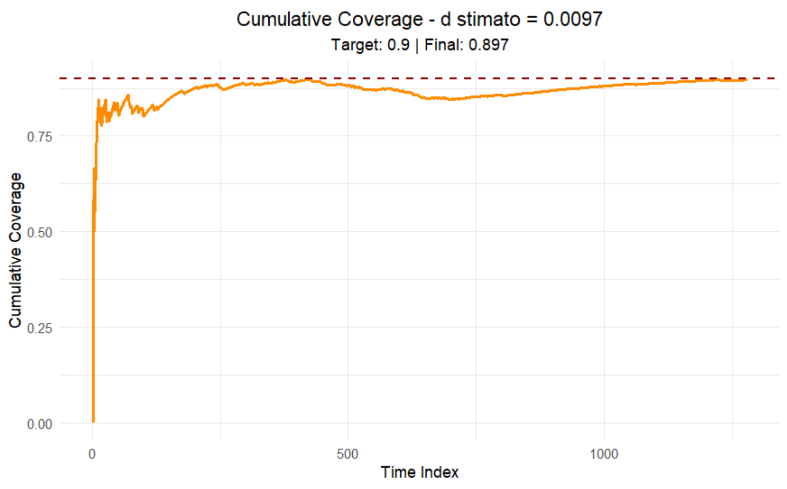

Figura 4.20. SP 500 Cumulative Coverage

With this graph, we obtain an important insight into the model's predictive capacity, we can observe the orange line representing the prediction's cumulative coverage compared to the target set of 90%, represented from the red dashed line. It is important to observe that, after an initial period which can be considered as adaptation, the model's coverage remains stable around a value of 0.897. This result proves that even if the model does not reach the target coverage, its coverage value is reasonable for use in financial markets, suggesting to us that predictions are reliable.

Figure 4.20 represents the evolution of the interval width; here, on the y-axis, are placed the prediction interval widths, while on the x-axis, the time is plotted. The green curve trend displays how the interval width changes in time, with an average width of 0.05253. Analyzing this graph is crucial for the investors since a larger interval width means that there is more uncertainty in our predictions. This analysis shows a certain degree of fluctuation, suggesting that market events such as crises or volatility peaks can influence the reliability of the predictions.

Turning to the overall analysis, Figure 4.22 compares the test data (blue lines) from the SP 500 index to the predictions obtained from the adaptive ARFIMA

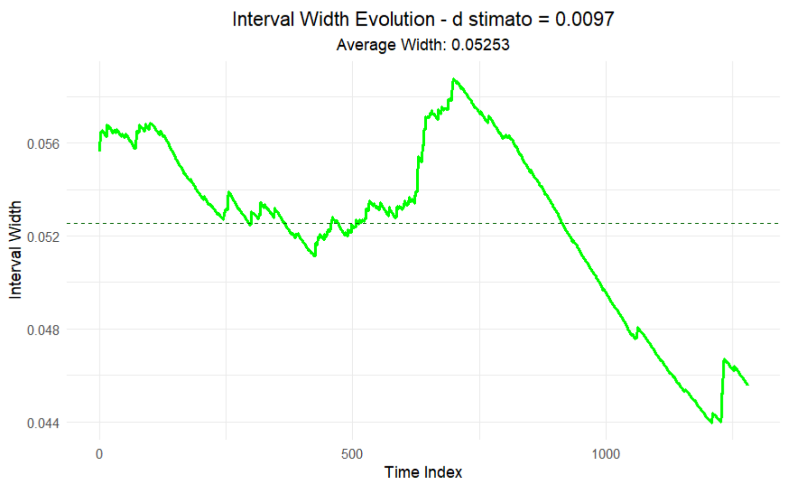

Figura 4.21. SP 500 Interval Width Evolution

model.

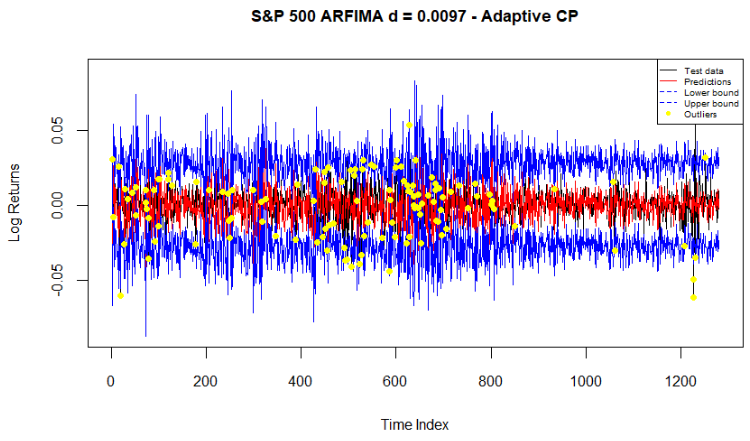

Figura 4.22. SP 500-Adaptive Conformal Prediction

The model's predictions are represented by the red lines, together with the upper and lower bounds (blue dashed lines and the outliers with the yellow dots. In this case, we can notice how predictions stay inside the uncertainty bound, even if some areas present inaccurate estimation, probably due to unexpected events such as financial crises, political changes, or even pandemics, which can cause significant fluctuations in the index returns.

In conclusion, the Adaptive Conformal Prediction analysis applied to the SP 500 returns not only enhances the understanding of long-term dynamics in market data but also offers practical insights to improve forecasting techniques.These results confirm the importance of adopting quantitative and analytical approaches in the finance field, where understanding how data behaves helps develop more efficient strategies in the future.

Capitolo 5

Conclusion

The primary objective of this thesis is to explore the application of the Conformal Prediction (CP) method, in both its standard and adaptive techniques, to long-memory processes.

This work aimed to fill an important methodological gap since traditional statistical approaches used for the prediction intervals construction have proven to be inadequate in the presence of persistence dependencies, typical of long memory.

For this reason, the objective posed in this thesis is to asses whether and how the Conformal Prediction can be a robust for modeling predictive uncertainty in this type of time series.

Throughout the paper, the main long-range dependence models have been presented: the aggregation of short memory processes, the Ising model, the Hierarchical variation model, and the model based on the Stochastic Partial Differential Equations (SPDEs).

Each of these models has provided a different perspective on the origins of long memory, underscoring that it can result from physical phenomena, collective interaction mechanisms, or physical constraints in space.

Afterwards, the study shifted to exploring the Conformal Prediction theoretical framework. Throughout the explanation of the two different techniques, the Split CP is based on the assumption of data exchangeability, and the Adaptive CP is suitable for non-stationary contexts as the financial markets, where data's distribution changes through time.

Both these methodologies provide us a coverage on the predictive intervals, without taking into account the error distribution, proving particularly suitable for analyses in which the independence or weak dependence hypothesis is not sustainable.

The empirical part of the paper proved the efficiency of the Conformal Prediction techniques applied to an ARFIMA (Autoregressive Fractionally Integrated Moving Average) simulated process, by changing the fractional differencing parameter d, which is essential for determining the degree of memory. It has been proved that there may also exist zones where the relationship between the long memory and the coverage results in a monotonic relationship, suggesting a more careful parameter calibration, as seen for the d=0.3 case.

Anyway, it is fundamental to analyze also the limits of this research. Firstly, the empirical analysis has been conducted on simulated data, and then it was implemented with two examples based on real data: the WTI crude oil prices and the SP 500 index.

This empirical study has proved that further verifications on more complex real datasets are useful to validate the generalizability of results. Moreover, the results' dependency on the goodness of the base model, which in our case is the ARFIMA model, suggests that the choice of the base learner is a critical point in any concrete application of the Conformal Prediction.

Looking forward, this paperwork clears the way for many extensions, for example, it can be extended to the integrations of conformal techniques with machine learning methods such as neural networks, to make predictions on more complex time series. In addition, the extension of the Conformal Prediction to spatial contexts as those described from the SPDE, may offer new prediction instruments useful in the pandemic field.

In conclusion, the Conformal Prediction can be considered the right methodology to deal with the challenges presented by the long-memory processes. Its ability to create valid predictive intervals without strong distributional assumptions makes it a robust but also adaptable alternative to other methodologies.

Bibliografia

Baillie, R. T. (1996). Long memory processes and fractional integration in econometrics. *Journal of econometrics*, 73(1):5–59.

Beran, J. (2017). *Statistics for long-memory processes.* Routledge.

Gibbs, I. and Candes, E. (2021). Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672.

Shen, T., Tao, Z., and Chen, H. (2024). Exploring long-memory process in the prediction of interval-valued financial time series and its application. *Journal of Systems Science and Complexity*, 37(2):759–775.

Tsay, R. S. (2005). *Analysis of financial time series.* John wiley & sons.

Zaffran, M., Féron, O., Goude, Y., Josse, J., and Dieuleveut, A. (2022). Adaptive conformal predictions for time series. In *International Conference on Machine Learning*, pages 25834–25866. PMLR.