UNIVERSIDADE FEDERAL DO PARANÁ

DOUGLAS TORRES DE CARVALHO

AVALIAÇÃO DO USO DE MODELOS GARCH PARA CRIPTOMOEDAS: UMA ANÁLISE DAS DEZ CRIPTOMOEDAS MAIS LÍQUIDAS ENTRE 2018 E 2022

CURITIBA

2024

AVALIAÇÃO DO USO DE MODELOS GARCH PARA CRIPTOMOEDAS: UMA ANÁLISE DAS DEZ CRIPTOMOEDAS MAIS LÍQUIDAS ENTRE 2018 E 2022

Dissertação apresentada ao curso de Pós-graduação Profissional em Economia, Setor de Economia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Economia.

Orientador: Prof. Dr. José Guilherme Silva Vieira

CURITIBA

2024

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIAS SOCIAIS APLICADAS

Carvalho, Douglas Torres de
Avaliação do uso de modelos Garch para criptomoedas : uma análise das dez criptomoedas mais líquidas entre 2018 e 2022 / Douglas Torres de Carvalho. – Curitiba, 2024.
1 recurso on-line : PDF.

Dissertação (Mestrado) – Universidade Federal do Paraná, Setor de Ciências Sociais Aplicadas, Programa de Pós-Graduação em Economia.
Orientador: Prof. Dr. José Guilherme Silva Vieira.

1. Criptomoedas. 2. Modelos estatísticos. 3. Análise de riscos. 4. Modelagem financeira. I. Vieira, José Guilherme Silva. II. Universidade Federal do Paraná. Programa de Pós-Graduação em Economia. III. Título.

Bibliotecária: Maria Lidiane Herculano Graciosa CRB-9/2008

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ECONOMIA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **DOUGLAS TORRES DE CARVALHO** intitulada: **Avaliação do uso de modelos Garch para criptomoedas: uma análise das dez criptomoedas mais líquidas entre 2018 e 2022**, sob orientação do Prof. Dr. JOSÉ GUILHERME SILVA VIEIRA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 03 de Julho de 2024.

Assinatura Eletrônica
15/07/2024 20:26:56.0
JOSÉ GUILHERME SILVA VIEIRA
Presidente da Banca Examinadora

Assinatura Eletrônica
03/07/2024 16:23:52.0
KEANU TELLES DA COSTA
Avaliador Externo (PIONTIFICIA UNIVERSIDADE CATÓLICA DO PARANÁ - PUC/PR)

Assinatura Eletrônica
03/07/2024 15:42:32.0
DAYANI CRIS DE AQUINO
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Dedicatória

Dedico esta dissertação a minha esposa Lahís, por todo o amor, companheirismo e incentivo para que eu pudesse desenvolver este projeto.

Dedico-o também a minha filha Malu, que me deu um novo sentido à vida, e que me proporciona diariamente uma alegria imensurável.

AGRADECIMENTOS

A minha esposa Lahís, aos meus pais Ricardo e Patrícia e meus irmãos Renan e Gabriela, por estarem sempre ao meu lado, me dando incentivo e apoio incondicional.

Ao meu orientador Prof. Dr. José Guilherme Silva Vieira, pelo seu apoio ao longo do curso, e suporte e correções durante a elaboração deste trabalho.

“The essence of investment management is the management of risks, not the management of returns.”

— Benjamin Graham

RESUMO

Este trabalho tem como objetivo identificar o modelo GARCH que melhor se aplica para cada um dos dez criptoativos estudados: Bitcoin, Ethereum, Theter, BNB, XRP, Cardano, Dogecoin, Tron, Litecoin e ChainLink, no período de janeiro de 2018 a dezembro de 2022. Para isso, foram utilizados modelos ARIMA para a média condicional e modelos GARCH de primeira ordem para a variância condicional. Diferentes modelos GARCH foram analisados, incluindo GARCH, EGARCH, GJRGARCH, Power ARCH (APARCH), Component GARCH (CSGARCH) e IGARCH, e o modelo ótimo foi escolhido com base nos critérios de informação AIC, BIC e RMSE. Os dados utilizados foram obtidos publicamente online, através da biblioteca crypto2 do software gratuito R, com base no site CoinMarketCap. Foram investigadas as estatísticas descritivas de cada um dos criptoativos analisados e testada a estacionariedade dos seus log-retornos. O modelo ARIMA que melhor se aplicou para cada um dos criptoativos foi definido e foram testados efeitos ARCH para cada um desses modelos ARIMA. Finalmente, foi determinado o modelo GARCH com melhor goodness-of-fit in-sample e out-of-sample. Os resultados mostraram que os modelos GARCH testados se aplicaram de forma diferente para cada um dos criptoativos estudados, indicando que é necessário modelar o comportamento dessas criptomoedas de forma individual.

Palavras-chave: Criptomoedas, GARCH, Análise de Risco, Modelagem Financeira, Volatilidade.

ABSTRACT

This work aims to identify the GARCH model that best fits each of the ten crypto assets studied: Bitcoin, Ethereum, Theter, BNB, XRP, Cardano, Dogecoin, Tron, Litecoin, and ChainLink, in the period from January 2018 to December 2022. ARIMA models were used for conditional mean and first order GARCH models for conditional variance. Different GARCH models were analyzed, including GARCH, EGARCH, GJRGARCH, Power ARCH (APARCH), Component GARCH (CSGARCH), and IGARCH, and the optimal model was chosen based on the information criteria AIC, BIC and RMSE. The data used was obtained publicly online through the crypto2 library of the free software R, based on the CoinMarketCap website. The descriptive statistics of each crypto were investigated, and the stationarity of their log returns was tested. The ARIMA model that best applied to each crypto asset was defined, and ARCH effects were tested for each of these ARIMA models. Finally, the GARCH model with the best goodness-of-fit was determined for in-sample and out-of-sample. The results showed that the GARCH models tested applied differently to each of the crypto assets studied, indicating that it is necessary to model the behavior of these cryptocurrencies individually.

Keywords: Cryptocurrencies, GARCH, Risk Analysis, Financial Modeling, Volatility.

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Σ - Somatório de números

α - alfa

δ - delta

ϵ - épsilon

LISTA DE ABREVIATURAS

ADF - Augmented Dickey–Fuller test

AIC - Akaike Information Criterion

ARIMA – Autorregressivo integrado de Média Móvel

GARCH - Heterocedásticos Condicional Autorregressivo Generalizado

USD – Dólar americano

BRL – Real brasileiro

SUMÁRIO

1 INTRODUÇÃO

O ano de 2022 foi marcado por uma relevante desvalorização no preço de diversas criptomoedas, de acordo com dados do CoinMarketCap (CoinMarketCap, 2023). O Bitcoin, criptomoeda mais conhecida pelo público em geral, caiu de um pico de aproximadamente USD 64.400 em novembro de 2021 para o patamar de aproximadamente USD 15.782 em novembro de 2022.

Ainda mais interessante o fato de que essa crise ocorreu na sequência de um boom verificado durante os anos de 2020 e 2021 e, do qual, se esperava uma persistência na performance positiva desses ativos. Esse comportamento de recorrentes picos e vales nos preços desses ativos indica a possibilidade de que exista uma alta volatilidade no universo das criptomoedas.

Nesse sentido, para fins de gerenciamento de risco de portfólio, seja individual ou corporativo, se faz necessário compreender e modelar o comportamento dessas criptomoedas. Assim, o presente trabalho tem como principal objetivo, analisar qual, dentre os modelos GARCH descritos na seção 4, possui o melhor goodness-of-fit[1] (AIC e BIC para a análise *in-sample* e RMSE para a análise *out-of-sample*) para os dados das dez criptomoedas aqui estudadas: Bitcoin, Ethereum, Theter, BNB, XRP, Cardano, Dogecoin, Tron, Litecoin e ChainLink para o período compreendido entre janeiro de 2018 e dezembro de 2022. Para o estudo *out-of-sample*, foi considerada a janela de 30 dias após o término dos dados.

Ainda, o estudo possui como objetivos específicos: (i) Investigar as estatísticas descritivas de cada um dos criptoativos analisados, (ii) Testar a estacionariedade do preço dos ativos e dos seus log-retornos, (iii) Definir o modelo ARIMA que mais se aplica para cada um dos criptoativos, (iv) Testar efeitos ARCH para cada um desses modelos ARIMA, (v) Determinar o modelo GARCH com melhor *goodness-of-fit in-sample,* (vi) Determinar o modelo GARCH com melhor *goodness-of-fit out-of-sample*.

Importante destacar que a escolha dessas criptomoedas seguiu critérios de liquidez e de negociação durante o período estudado. Foram consideradas criptomoedas aptas aquelas que possuíram negociação em todos os dias analisados,

---

[1] Goodness-of-fit: é um teste estatístico que tenta determinar se um conjunto de valores observados corresponde aos esperados no modelo aplicável.

de modo que os dados pudessem ter continuidade, visto que o trabalho se concentra na volatilidade entre dias (*interday*). Na sequência, as criptomoedas foram ranqueadas com base no valor de mercado cap de fechamento do período.

Aqui cabe destacar que o valor de mercado médio do período poderia nos trazer uma combinação diferente de moedas, principalmente considerando aquelas posicionadas nas últimas posições ranqueadas. Entretanto, isso absolutamente não invalida ou traz prejuízo ao estudo, na medida em que a análise das criptomoedas foi feita de forma independente entre elas. Uma nova combinação poderia nos trazer informações novas sobre outras criptomoedas, mas não alterariam o resultado das criptos aqui estudadas.

A justificativa para sustentar o presente projeto de pesquisa reside no fato de que elas são de interesse de grande parte do público que investe seus recursos usando *fintechs.* Assim, entender o comportamento desses ativos tem um valor especial, na medida em que existe um grande potencial para que sejam fonte relevante de receita para essas empresas.

Além disso, pelas criptomoedas serem uma temática atual e, alguns estudos anteriores terem analisado o comportamento desses ativos sob um prisma de volatilidade muito distinto do que se observou nos últimos anos, o presente estudo possibilita ampliar o debate para uma janela temporal onde o volume negociado é muito mais representativo. Boa parte dos estudos sobre o tema foram feitos entre 2011 e 2017. Nesse contexto (i) o preço desses ativos aparenta ser menos volátil nesse período e (ii) o volume negociado, é uma fração do que se tornou o mercado a partir de 2018.

Portanto, resgatar a temática sobre uso de modelos GARCH para criptomoedas, dentro dessa nova conjuntura de mercado e progresso na adoção desses ativos pelo mercado, se justifica não só pelo contexto profissional, bem como pelo potencial de trazer um maior debate acadêmico acerca do tema.

Para isso, a metodologia, que será aprofundada no capítulo 3, consiste na captura do fechamento diário do preço de cada um dos criptoativos e, na utilização e comparação de modelos ARIMA e GARCH. Sendo assim, o modelo ótimo será escolhido com base na comparação entre os critérios de informação (AIC e BIC para *in-sample* e RMSE para *out-of-sample*).

2 REVISÃO DE LITERATURA

O capítulo em questão tem por objetivo trazer uma revisão da literatura dos principais pontos que serão utilizados no trabalho. Primeiro, o item 2.1 tem por objetivo indicar os principais trabalhos que serviram como base para o presente estudo. Nesse contexto, serão indicados os objetivos e resultados de cada um desses estudos.

O item 2.2 tem por finalidade apresentar uma breve explicação sobre retornos e volatilidade, contextualizando o leitor acerca da base do ponto de vista de finanças em que o trabalho se encontra.

Os itens 2.3 e 2.4 abordam a revisão literária sobre os modelos econométricos utilizados no estudo. Ainda que um leitor que não tenha tanta afinidade ao assunto consiga entender os resultados do estudo sem grandes dificuldades, ambos os itens são importantes para entendimento da base econométrica utilizada.

Já o item 2.5 tem como propósito indicar a fundamentação teórica para a seleção dos modelos, tanto *in-sample* quanto *out-of-sample*.

Por fim, o item 2.6 busca dar um breve contexto sobre cada uma das criptomoedas aqui estudadas. Ainda que o objetivo não seja aprofundar em temas técnicos de cada cripto, o item certamente ajudará ao leitor a se situar e compreender melhor a proposta de cada uma dessas moedas.

2.1 TRABALHOS ANTERIORES

O trabalho traz uma análise sobre o comportamento de dez criptomoedas através de modelos de séries temporais. Alguns trabalhos analisaram esse mesmo problema através de óticas um pouco distintas e que serão abordadas neste capítulo.

P. Katsiampa (2017) e L. Ø. Bergsli, A. F. Lind, P. Molnár e M. Polasik (2021) tiveram como objetivo, indicar o melhor modelo para análise da volatilidade do Bitcoin.

No trabalho de Katsiampa, intitulado "*Volatility estimation for Bitcoin: A comparison of GARCH models*", o autor utiliza-se dos log-retornos do fechamento do bitcoin para o período entre julho/2010 e outubro/2016. O trabalho foi um dos primeiros a apresentar uma comparação entre diferentes modelos do tipo GARCH. Katsiampa utilizou-se dos modelos GARCH, EGARCH, TGARCH, Asymetric Power ARCH (APARCH), Component GARCH (CSGARCH) e Asymetric Component GARCH (ACGARCH).

Importante destacar que o estudo leva em consideração apenas o *goodness of fit in-sample*. Nesse sentido, o resultado obtido por Katsiampa é o de que o Component GARCH (CSGARCH) é aquele que explica melhor o comportamento dessa criptomoeda no período estudado.

O segundo trabalho, de L. Ø. Bergsli, A. F. Lind, P. Molnár e M. Polasik, intitulado “Forecasting volatility of Bitcoin”, teve como foco estudar qual modelo é o mais adequado para prever a volatilidade do Bitcoin e utilizou-se dos log-retornos para o período entre setembro/11 e setembro/18. A diferença em relação ao trabalho anterior se deu na medida em que além de serem considerados vários GARCH, também foram introduzidos dois modelos HAR (*heterogeneous autoregressive*). Importante destacar que esse estudo teve como diferencial, além da introdução de outro modelo, o uso de dados de alta frequência, em comparação aos dados diários e a comparação dos modelos *out-of-sample*, para os períodos de um, dois, cinco, dez e quinze dias.

Como resultado, chegou-se à conclusão de que os modelos EGARCH e APARCH apresentam melhor desempenho entre os modelos GARCH. Além disso, que modelos HAR baseados na variância realizada têm melhor desempenho do que os modelos GARCH baseados em dados diários. Ou seja, quanto mais de curto prazo for a análise, melhor os modelos HAR se comportariam.

O trabalho de A. Bashar, A. Muneer e A. Ammar. (2021). intitulado “Performance of ARCH and GARCH Models in Forecasting Cryptocurrency Market Volatility” teve como objetivo investigar a heterocedasticidade condicional autorregressiva (ARCH) e a heterocedasticidade condicional autorregressiva generalizada (GARCH) na previsão de nove criptomoedas (Bitcoin, Ethereum, Ripple, Litecoin, Ethereum classic, Link, EOS, USDT e BCH). Além disso, tem semelhança com o presente trabalho na medida em que se buscou um período mais recente de tempo, sendo considerado um intervalo de dados diários de 2010 a 2020.

Os resultados mostram que existe a presença de efeitos ARCH e, o modelo GARCH têm um efeito significativo na previsão da volatilidade do mercado de criptomoedas, o que significa que a volatilidade passada das criptomoedas afeta a volatilidade atual delas.

Outro trabalho que procurou estudar o tema de criptomoedas e modelos GARCH foi o de Kyei-Mensah, J (2023) intitulado “Asymmetric Reverting and Volatility in Cryptocurrency Markets”. O estudo empregou o modelo ANST-GARCH para

investigar as respostas assimétricas de reversão a média e volatilidade para 17 criptomoedas (bitcoin (BTC), ethereum (ETH), litecoin (LTC), nem (XEM), dash (DASH), dogecoin (DOGE), monero (XMR), vertcoin (VTC), verge (XVG), digibyte (DGB), Stellar (XLM), tether (USDT), maidsafecoin (MAID), ripple (XRP), bitcoin cash (DCH), cardano (ADA), and siacoin (SC)).

O estudo encontrou fortes evidências para o modelo ANST-GARCH, bem como assimetrias consideráveis em relação ao comportamento de investidores, que reagem exageradamente a certas notícias do mercado de criptomoedas, além de revelar que a volatilidade assimétrica é significativa para estes tipos de ativos.

Os trabalhos descritos acima, apesar de tratarem temas relacionados ao objeto de estudo do presente trabalho, se diferenciam do que é proposto neste estudo principalmente pelo aspecto restritivo: (i) em termos de escolha de criptomoedas, onde tanto o trabalho de Katsiampa (2017) como o de Bergsli, Lind, Molnár e Polasik (2021), abordaram apenas uma criptomoeda. E (ii) em termo de modelos, onde tanto o trabalho de Bashar et. al (2021) como o de J. Kyei-Mensah se utilizam apenas de um modelo GARCH para análise da volatilidade dos ativos, sendo o primeiro um modelo GARCH e o segundo um modelo ANST-GARCH.

Além destes, o trabalho que mais se aproxima do que é proposto nesta investigação, é o trabalho de Jefrey Chu, Stephen Chan, Saralees Nadarajah e Joerg Osterrieder (2017). O trabalho, intitulado “GARCH modeling of cryptocurrencies”, tem por objetivo estimar o modelo que melhor se adequa para as sete criptomoedas com maior Market Cap, de acordo com o site CoinMarketCap.

O estudo buscou analisar o *goodness of fit* de doze modelos GARCH, nomeadamente SGARCH, EGARCH, GJRGARCH, APARCH, IGARCH, CSGARCH, GARCH, TGARCH, AVGARCH, NGARCH, NAGARCH para os log-retornos das sete criptomoedas. Importante destacar que esse estudo também teve por objetivo examinar diferentes distribuições para cada modelo

O resultado obtido foi: (i) do ponto de vista de distribuição, apontou-se que a distribuição normal apresentava os menores resultados de AIC e BIC para todos os modelos e criptomoedas, com exceção apenas do modelo TGARCH e AVGARCH para a moeda XPR, onde a distribuição normal assimétrica (*skew normal*) apresentou melhor resultado do que a distribuição normal. (ii) Dentre os doze modelos GARCH com maior *goodness of fit*, o modelo IGARCH apresenta os melhores resultados para Bitcoin, Dash, Litecoin, Maidsafecoin e Monero. Já o modelo GJRGARCH apresenta

os melhores valores para Dogecoin. Por fim, o modelo GARCH convencional é o que possui o melhor fit para o Ripple (XRP).

Nesse contexto, o presente trabalho se diferencia do trabalho acima: (i) pelo uso de criptomoedas não incluídas no estudo indicado. Sendo Bitcoin, Litecoin, Dogecoin e XRP as únicas presentes em ambos os trabalhos; (ii) pela atualização dos dados, na medida em que o trabalho em questão possui dados apenas até maio/2017.

Como indicado anteriormente, o segundo item acima possui uma grande relevância, na medida em que os criptoativos, destacadamente as criptomoedas, são ativos relativamente novos e que estão sendo cada vez mais utilizados. Podemos destacar, portanto, que os diferenciais para os trabalhos acima são: (i) o preço desses ativos aparenta ser menos volátil nesse período e, (ii) o volume negociado no período é uma fração do que se tornou o mercado a partir de 2018, conforme podemos observar no gráfico 1 abaixo (para o caso do Bitcoin), onde tanto o preço (em verde), quanto o volume (em cinza) no período de ambos os estudos, são uma fração do que é hoje o mercado para essa criptomoeda.

GRÁFICO 1 – Bitcoin CoinMarketCap

FONTE: CoinMarketCap (2023).

2.2 RETORNOS E VOLATILIDADE

De acordo com Morettin (2016), um dos objetivos em finanças é a avaliação de riscos de uma carteira de ativos financeiros. Sendo o risco comumente medido em termos de variação dos preços dos ativos.

Essa variação pode ser denotada pelas fórmulas abaixo:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \quad (1)$$

$$r_t = log\,\frac{P_t}{P_{t-1}} \quad (2)$$

Onde:

R_t são os retornos simples do ativo

P_t é o preço no tempo t

P_{t-1} é o preço no tempo t-1

r_t são os log-retornos do ativo

É, portanto, preferível trabalhar com retornos na medida em que esses são livres de escala e possuem propriedades estatísticas gerais mais interessantes, como de estacionariedade e ergodicidade.

O presente trabalhou optou por utilizar os log-retornos dado que: (i) no estudo, assumimos que os preços são distribuídos log normalmente (nas considerações finais, indicamos como possibilidade de continuação do estudo a inclusão de outras distribuições de probabilidade); (ii) os log-retornos são aditivos ao longo do tempo. Desse modo, a soma dos log-retornos de períodos consecutivos é o log-retorno do período total.

No que diz respeito a volatilidade, Morettin a entende como o desvio padrão condicional de um retorno. E, ainda que não seja medida diretamente, ela pode se manifestar de diferentes maneiras em uma séria financeira. Há, segundo ele, três enfoques para o cálculo da volatilidade:

(i) Equacionar um preço de mercado observável com o preço modelo de uma opção. Nesse caso, obtemos a *volatilidade implícita*, usualmente atrelada a fórmula de Black-Scholes.

(ii) Modelar diretamente a volatilidade da série de retornos através de modelos que capturam essa variável, como modelos ARCH. Com isso, é obtida a *volatilidade estatística*.

(iii) Modelar a volatilidade por meio da média de uma função dos últimos k retornos. A partir disso é obtida a *volatilidade histórica*.

Outras abordagens podem ser utilizadas usando-se uma combinação dos enfoques acima indicados. Nesse trabalho, a volatilidade será tratada como uma volatilidade estatística, sendo modelada através de modelos econométricos, mais especificamente, os modelos GARCH.

2.3 MODELOS ECONOMÉTRICOS (ARIMA E GARCH)

Box e Jenkins (1976) introduziram pela primeira vez os modelos ARIMA, cujo termo deriva de:

AR = autoregressivo;

I = integrado; e

MA = média móvel

De acordo com Asteriou e Hall (2021), o modelo de série temporal mais simples é o modelo autoregressivo de ordem um, ou modelo AR(1).

$$Y_t = \varphi Y_{t-1} + u_t \ (3)$$

Onde:

Y_t são os valores da série temporal em t

φ é o parâmetro autorregressivo, que mede a força da relação entre o valor atual e o valor anterior.

Y_{t-1} são os valores da série temporal em t-1

u_t é o termo de erro ou ruído branco, representando o componente aleatório.

e, por simplicidade, não incluímos uma constante, φ <1. A suposição por trás do modelo AR(1) é que o comportamento da série temporal de Y_t é, em grande parte, determinado pelo seu próprio valor no período anterior. Portanto, o que acontecerá em t depende em grande parte do que aconteceu em t−1. Alternativamente, o que acontecerá em t+1 será determinado pelo comportamento da série no tempo atual t.

Já o modelo de média móvel mais simples é o de ordem um, ou modelo MA(1), que tem a forma:

$$Y_t = u_t + \theta u_{t-1} \quad (4)$$

Onde:

Y_t são os valores da série temporal em t

θ é o parâmetro de média móvel, que mede o impacto dos erros passados no valor atual.

u_{t-1} O termo de erro ou ruído branco no tempo t-1

u_t O termo de erro ou ruído branco no tempo t

Assim, a implicação por trás do modelo MA(1) é que Y_t depende do valor do erro passado imediato, que é conhecido no tempo t.

Depois de apresentar os processos AR e MA, podemos criar combinações dos dois processos para dar uma nova série de modelos chamados modelos ARMA(p, q).A forma geral do ARMA model é um modelo ARMA(p, q) é da forma:

$$Y_t = \sum_{i=1}^{p} \varphi_i Y_{t-1} + u_t + \sum_{j=1}^{q} \theta_j u_{t-j} \quad (5)$$

Onde:

Y_t são os valores da série temporal em t

Y_{t-1} são os valores da série temporal em t-1

φ são os parâmetros autorregressivos, onde i = 1, 2, ..., p. Esses parâmetros capturam a dependência do valor atual em relação aos valores anteriores da série temporal.

θ São os parâmetros de média móvel, onde j = 1, 2, ..., q. Esses parâmetros capturam a dependência do valor atual em relação aos erros passados.

u_t O termo de erro ou ruído branco no tempo t

u_{t-j} O termo de erro ou ruído branco no tempo t-j

Os modelos ARMA (p,q) só podem ser feitos com séries temporais estacionárias. Isso significa que a média, a variância e a covariância da série são constantes ao longo do tempo. Contudo, a maioria das séries temporais financeiras mostram tendências ao longo do tempo e, portanto, possuem média diferente ao longo do tempo, o que indica que as séries são não estacionárias. Para evitar esse problema e induzir a estacionariedade, precisamos passar por um processo chamado diferenciação (d). Através desse processo, temos o modelo ARIMA(p,d,q)

O primeiro modelo que fornece uma estrutura sistemática para modelagem de volatilidade é o modelo ARCH de Engle (1982). A ideia básica dos modelos ARCH é que: (a) o choque dos retornos de um ativo é serialmente não correlacionado, mas dependente, e (b) a dependência dos valores pode ser descrita por uma função quadrática simples de seus valores defasados.

Especificamente, um modelo ARCH(m) assume que:

$$a_t = \sigma_t \in_t, \qquad \sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2 \quad (6)$$

Onde $\in_t$ é uma sequência de valores aleatórios independentes e identicamente distribuídos com média zero e variância 1, $\alpha_0 > 0$ e $\alpha_i \geq 0$ para i > 0. Os coeficientes α_i devem satisfazer algumas condições de regularidade para garantir que a variância incondicional de a_t é finita. Na prática, assume-se frequentemente que $\in_t$ segue o padrão normal.

A partir da estrutura do modelo, vê-se que grandes choques quadráticos passados $\{a_{t-i}^2\}_{i=1}^m$ implicam uma grande variância condicional σ_t^2 para a inovação em a_t. Consequentemente, os valores de a tendem a assumir um valor grande (em módulo). Isso significa que, sob a estrutura ARCH, grandes choques tendem a ser seguidos por outro grande choque.

Bollerslev (1986), propôs uma extensão útil ao modelo ARCH, conhecida como ARCH generalizado (GARCH). O modelo GARCH é especificado conforme segue abaixo:

$$y_t = \mu_t + u_t \quad (7)$$

$$\sigma^2_t = \omega + \alpha u^2_{t-1} + \beta \sigma^2_{t-1} \quad (8)$$

$$u_t = \sigma_t \in_t, \qquad \in_t \sim N(0,1)$$

Onde:

Y_t são os valores da série temporal no tempo t.

μ_t é a média condicional de Y_t no tempo t.

u_t é o termo de erro ou inovação no tempo t. É uma variável aleatória com média zero e variância condicional σt^2.

σt^2 é a variância condicional de Y_t no tempo t. Representa a variância dos retornos, condicional à informação disponível até o tempo t-1.

ω Constante não negativa que representa a variância incondicional de longo prazo.

α Parâmetro não negativo que mede o impacto dos choques passados (u^2_{t-1}) na variância atual.

β Parâmetro não negativo que mede a persistência da volatilidade, ou seja, o impacto da variância passada (σ^2_{t-1}) na variância atual.

u^2_{t-1} Quadrado do termo de erro no período anterior. Captura o impacto de choques passados na variância atual.

O modelo assume que os retornos são heterocedásticos e o processo de variância pode ser previsto. A equação de variância, de fato, nos diz que a variância condicional de t+1 é igual à média ponderada da variância de longo prazo.

2.4 EXTENSÕES DO MODELO GARCH

O modelo exponencial GARCH (EGARCH) foi desenvolvido pela primeira vez por Nelson (1991), e a equação de variância para este modelo é dada por:

$$\log(h_t) = \gamma + \sum_{j=1}^{q} \zeta_j \left|\frac{u_{t-j}}{\sqrt{h_{t-j}}}\right| + \sum_{j=1}^{q} \xi_j \frac{u_{t-j}}{\sqrt{h_{t-j}}} + \sum_{i=1}^{p} \delta_i \log(h_{t-i}) \quad (9)$$

Onde:

h_t são os valores da variância condicional no tempo t.

γ Um termo constante que captura o nível de volatilidade de longo prazo.

$\sum_{j=1}^{q} \zeta_j \left|\frac{u_{t-j}}{\sqrt{h_{t-j}}}\right|$ o primeiro termo captura o efeito assimétrico dos choques na volatilidade.

$\sum_{j=1}^{q} \xi_j \frac{u_{t-j}}{\sqrt{h_{t-j}}}$ o segundo termo captura o efeito simétrico dos choques na volatilidade.

$\sum_{i=1}^{p} \delta_i \log(h_{t-i})$ o terceiro termo captura a persistência da volatilidade.

Onde γ, ζ, ξ e δ são parâmetros a serem estimados. Observe que o lado esquerdo é o logaritmo da série de variância. O modelo EGARCH permite testar assimetrias assim como o TGARCH. Para testar assimetrias, os parâmetros importantes são os ξs. Se ξ1 = ξ2 = ⋯ = 0, então o modelo é simétrico. Quando ξj < 0, então os choques positivos (boas notícias) geram menos volatilidade do que os choques negativos (más notícias).

O modelo GJR GARCH de Glosten et al. (1993) modela choques positivos e negativos na variância condicional assimetricamente, através do uso da função indicadora I,

$$\sigma^2 = (\omega + \sum_{j=1}^{m} \zeta_j \upsilon_{jt}) + \sum_{j=1}^{q} (\alpha_j \varepsilon^2_{t-j} + \gamma_j I_{t-j} \varepsilon^2_{t-j}) + \sum_{j=1}^{p} \beta_j \sigma^2_{t-j} \quad (10)$$

Onde:

σ^2 são os valores da variância condicional no tempo t.

ω Um termo constante que captura o nível de volatilidade de longo prazo.

$\sum_{j=1}^{m} \zeta_j \, \text{ɣ}$ o primeiro termo captura o impacto de inovações passadas na variância condicional. Em que υ_{jt} é igual a 1 se ε_{t-j} for negativo e 0 caso contrário.

α_j Parâmetro que captura o impacto de inovações quadradas positivas passadas na variância condicional

γ_j Parâmetro que captura o impacto de inovações quadradas negativas passadas na variância condicional

I_{t-j} Função indicadora que é igual a 1 se ε_{t-j} for negativo e 0 caso contrário

ε_{t-j} Inovações passadas

β_j Parâmetro que captura a persistência da volatilidade

Devido à presença da função indicadora, a persistência do modelo depende da assimetria da distribuição condicional utilizada.

O modelo ARCH de potência assimétrica de Ding et al. (1993) permite tanto a alavancagem quanto um efeito Taylor, em homenagem a Taylor (1986) que observou que a autocorrelação amostral dos retornos absolutos era, geralmente, maior do que os retornos quadrados.

$$\sigma_t^{\delta} = (\omega + \sum_{j=1}^{m} \zeta_j v_{jt}) + \sum_{j=1}^{q} \alpha_j (|\varepsilon_{t-j}| + \gamma_j \varepsilon_{t-j})^{\delta} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^{\delta} \quad (11)$$

Onde:

σ_t^{δ} é a variância condicional no tempo t

ω é uma constante que representa a variância incondicional de longo prazo

ζ_j parâmetro que captura o impacto dos choques passados na variância condicional

v_{jt} é uma variável dummy que indica se o choque passado foi positivo ou negativo

α_j parâmetro que captura o impacto dos quadrados dos choques passados na variância condicional

γ_j parâmetro que captura o impacto de inovações quadradas negativas passadas na variância condicional

β_j parâmetro que capturam a persistência da volatilidade

Vários submodelos surgem deste modelo:

• O modelo GARCH simples de Bollerslev (1986) quando $\delta = 2$ e $\gamma_j = 0$.

• O modelo GJR GARCH de Glosten et al. (1993) quando $\delta = 2$.

• O modelo Threshold GARCH (TGARCH) de Zakoian (1994) quando $\delta = 1$.

O modelo CSGARCH de Lee e Engle (1999) decompõe a variância condicional em uma variação permanente e um componente transitório, de modo a investigar os

movimentos de volatilidade de longo e curto prazo afetando os ativos, o modelo pode ser escrito como:

$$\sigma_t^2 = q_t + \sum_{j=1}^{q} \alpha_j(\varepsilon_{t-j}^2 - q_{t-j}) + \sum_{j=1}^{p} \beta_j(\sigma_{t-j}^2 - q_{t-j}) \quad (12)$$

$$q_t = \omega + pq_{t-1} + \phi(\varepsilon_{t-1}^2 - \sigma_{t-1}^2) \quad (13)$$

Onde:

σ_t^2 Variância condicional no tempo t

q_t Componente permanente da variância condicional no tempo t

ε_{t-j}^2 Resíduo no tempo t-j

α_j Parâmetro que captura o impacto dos quadrados dos resíduos passados na componente transitória

β_j Parâmetro que captura a persistência da componente transitória

ω Constante que representa o nível de longo prazo da componente permanente

p Parâmetro autorregressivo que captura a persistência da componente permanente

ϕ Função não linear que captura a assimetria e a não linearidade na relação entre os choques e a componente permanente

onde a interceptação do modelo GARCH agora varia no tempo seguindo a primeira ordem dinâmica de tipo autorregressivo. A diferença entre a variância condicional e sua tendência, $\sigma_{t-j}^2 - q_{t-j}$ é o componente transitório da variância condicional.

2.5 CRITÉRIOS DE INFORMAÇÃO

O critério de informação de Akaike (AIC) é um índice de ajuste relativo baseado em critérios de informação. Foi desenvolvido como uma aproximação da precisão preditiva fora da amostra de um modelo, de acordo com os dados disponíveis (Akaike, 1974). Esta medida utiliza o log-verossimilhança, mas adiciona um termo penalizador associado ao número de variáveis, dado que é sabido que adicionando variáveis pode-se melhorar o ajuste dos modelos. Assim, a AIC tenta equilibrar o *goodness of fit* versus a inclusão de variáveis no modelo.

O AIC é calculado da seguinte forma:

$$AIC = -2\frac{\ell}{n} + 2\frac{k}{n} \quad (14)$$

Onde:

n quantidade de observações

k número de parâmetros (regressores e intercepto)

ℓ a função de log verossimilhança, que é calculada pela equação:

$$\ell = -\frac{n}{2}(1 + \ln(2\pi) + \ln(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2) \quad (15)$$

O critério de informação Bayesiano (BIC) é outra medida estatística para a avaliação comparativa entre modelos de séries temporais. Foi desenvolvido pelo estatístico Gideon Schwarz e está intimamente relacionado com a AIC. A diferença entre BIC e AIC se manifesta quando adicionamos uma série de k parâmetros (regressores e/ou interceptos), a fim de aumentar a qualidade de ajuste do modelo. Nesse caso, o BIC penaliza mais (em comparação com o AIC) esse aumento de parâmetros, conforme segue abaixo:

$$BIC = -2\frac{\ell}{n} + 2\frac{k * \ln(n)}{n} \quad (16)$$

Onde:

n quantidade de observações

k número de parâmetros (regressores e intercepto)

ℓ a função de log verossimilhança, que é calculada pela equação:

$$\ell = -\frac{n}{2}(1 + \ln(2\pi) + \ln(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2) \quad (17)$$

O RMSE é um método para testar se um modelo é preciso ao fazer previsões futuras. O RMSE é extraído através da raiz quadrada dos erros quadráticos médios de todos os resíduos. Ela fornece informações sobre o desempenho de curto prazo de um modelo, permitindo uma comparação termo a termo da diferença real entre o valor estimado e o valor medido. Quanto menor o valor, melhor o desempenho do modelo. Uma desvantagem deste teste é que poucos erros grandes na soma podem produzir um aumento significativo no RMSE. Pode ser calculado através da fórmula abaixo:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{p,i} - Y_{a,i})^2} \quad (18)$$

onde

$Y_{p,i}$ são os outputs previstos

$Y_{a,i}$ são os outputs reais

Espera-se que os erros sejam mínimos para um bom modelo de previsão.

2.6 CRIPTOMOEDAS

De acordo com o Fundo Monetário Internacional (FMI, 2024), as criptomoedas são moedas que se baseiam na tecnologia blockchain, para construir um registro (efetivamente uma base de dados) que é mantido através de uma rede. Para garantir que a mesma criptomoeda não seja gasta duas vezes, cada membro da rede verifica e valida cada uma das transações utilizando criptografia.

Uma vez alcançado um consenso descentralizado entre os membros da rede, a transação é adicionada à uma espécie de livro-razão, onde é validada. Esse livro-razão fornece um histórico completo das transações associadas a uma criptomoeda específica, sendo permanente e não manipulável por uma única entidade da cadeia.

Nesse contexto, este item tem por objetivo trazer uma breve explicação sobre cada uma das criptomoedas estudadas aqui. Não se tem por objetivo entrar nos detalhes técnicos dessas moedas, mas sim trazer um pouco de contexto sobre criação e principal uso de cada uma delas.

Bitcoin: De acordo com o site oficial da moeda (bitcoin.org, 2024), o Bitcoin é uma rede que funciona de forma consensual, onde foi possível criar uma forma de pagamento e uma nova moeda completamente digital. É a primeira rede de pagamento descentralizada (ponto-a-ponto) onde os usuários é que gerenciam o sistema, sem necessidade de intermediador ou autoridade central. Da perspectiva do usuário, Bitcoin funciona como dinheiro para a Internet. Bitcoin também pode ser visto como o mais promissor sistema de contabilidade de entrada tripla existente.

Ethereum: De acordo com o site oficial da moeda (ethereum.org, 2024), a Ethereum é uma rede de computadores em todo o mundo que seguem um conjunto de regras designado por protocolo Ethereum. A rede Ethereum funciona como a base para comunidades, aplicações, organizações e ativos digitais que qualquer pessoa pode construir e utilizar. A sua criptomoeda, o ETH é segundo a sua organização, a força vital da rede Ethereum. Sempre que um ETH é enviado, o utilizador paga uma taxa em ETH para utilizar a rede Ethereum.

Theter: é a maior stablecoin em capitalização de mercado. As stablecoins, como o Tether, são usadas para fazer transferências entre diferentes criptomoedas ou para transferir seus investimentos para dentro ou fora de moedas fiduciárias. O valor do Theter está indexado ao dólar americano. Em teoria, isso significa que o Tether não deve ser afetado pela volatilidade que pode impactar dramaticamente os valores de outras criptomoedas, como o Bitcoin (BTC).

BNB (ou Binance Coin): foi lançado por meio de uma Oferta Inicial de Moedas (ou ICO) em 2017, pouco antes criação da Binance Exchange. Apesar de estar diretamente ligado a uma exchange, o BNB atua como uma criptomoeda padrão. O BNB foi originalmente emitido como um token, rodando na rede Ethereum. Após um tempo, ocorreu um swap na proporção 1:1 e a moeda foi transferida para a Binance Chain, rede similar a Ethereum da Binance Exchange.

XRP: funciona de forma muito similar ao BNB, sendo que no caso do XRP a empresa que está por trás é a Ripple, que fornece tanto o protocolo (Ripple Chain), quanto a exchange.

Cardano: é uma plataforma blockchain, e possui um viés educacional e de pesquisa sobre a tecnologia sendo, segundo a plataforma, a primeira a ser baseada em pesquisas revisadas por pares e desenvolvida por meio de métodos baseados em evidências. A moeda da blockchain, também conhecida pelo código ADA.

Dogecoin: é uma criptomoeda que pode ser utilizada como meio de pagamento. É um fork[2] do Litecoin, ou seja, uma atualização feita no protocolo. Além disso, seu código é baseado no Bitcoin, a criptomoeda mais famosa do planeta até o momento.

Tron: usa os recursos da tecnologia de rede blockchain e peer-to-peer (P2P) para eliminar o intermediário e permitir que criadores de conteúdo vendam seu trabalho diretamente aos consumidores. A moeda usada na rede é conhecida pela sigla TRX. Os usuários da rede usam o TRX para pagar diretamente aos criadores de conteúdo para acessar seus aplicativos.

Litecoin: também conhecida pela sigla (LTC), é uma criptomoeda criada por um ex-engenheiro do Google. Ela foi adaptada do código-fonte aberto do Bitcoin, mas

---

[2] Um fork de criptomoeda é uma atualização de software blockchain que pode implementar pequenas alterações no protocolo existente ou fazer com que ele seja dividido em dois protocolos separados e incompatíveis

com diversas modificações, sendo, portanto, uma fork do Bitcoin. Tem como diferenciais aspectos como taxa de geração de blocos mais rápida e uso de Scrypt como esquema de prova de trabalho.

ChainLink (LINK): é uma plataforma de criptomoeda e tecnologia que permite que empresas não-blockchain se conectem com segurança a plataformas blockchain. Chainlink usa tecnologia blockchain para permitir cálculos dentro e fora da cadeia com segurança, suportando o que chama de contratos inteligentes híbridos e seu protocolo de interoperabilidade entre cadeias. O blockchain Chainlink está hospedado na plataforma Ethereum.

3 METODOLOGIA E DADOS

Os dados usados neste trabalho são os preços de fechamento diários das dez criptomoedas a seguir: Bitcoin, Ethereum, Theter, BNB (Binance Coin), XRP (Ripple), Cardano, Dogecoin, Tron, Litecoin e ChainLink, de 01 de janeiro de 2018 a 31 de dezembro de 2022, o que corresponde a um total de 1825 observações para cada cripto. Os dados estão disponíveis publicamente online, através da biblioteca crypto2 do software gratuito R via web API (os dados podem ser acessados através do código, disponibilizado no Apêndice). Além disso, é importante destacar que esses dados gerados na biblioteca crypto2, tem como base o site CoinMarketCap e podem ser verificados através do site: https://coinmarketcap.com.

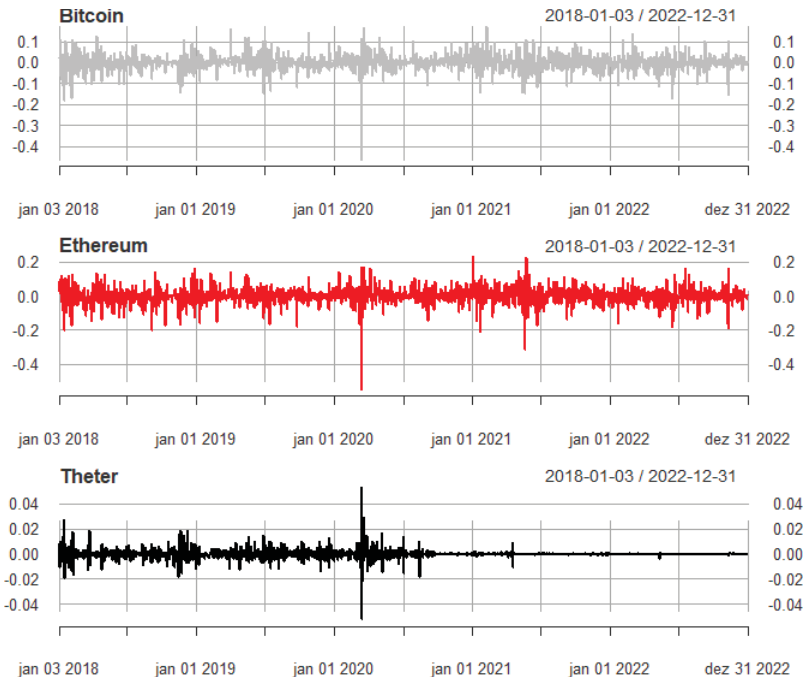
A escolha de utilizar dados do CoinMarketCap, em vez de dados de dados de diferentes exchanges, deve-se à dois pontos: (i) o CoinMarketCap já calcula os seus preços de fechamento com base na média ponderada pelo volume de preços de diferentes exchanges. Em outras palavras, o resultado já é uma boa proxy do que seria o resultado obtido tomando-se diferente exchanges diretamente e; (ii) para garantir uma melhor qualidade e confiabilidade dos dados, tomou-se como critério a utilização de uma fonte única de dados de modo a não criar uma possível quebra de continuidade nos dados, na medida em que algumas exchanges poderiam ter fechado no período ou, o ativo não ter sua cotação em determinado dia em uma dessas exchanges.

Outro ponto importante resultante dessa escolha é a de replicabilidade do trabalho, uma única fonte de consulta irá facilitar futuros estudos que busquem replicar parte do presente ensaio.

Os retornos dos ativos são calculados tomando a log-diferença entre dois preços consecutivos e podem ser observados nos gráficos 2 a 4 abaixo[3].

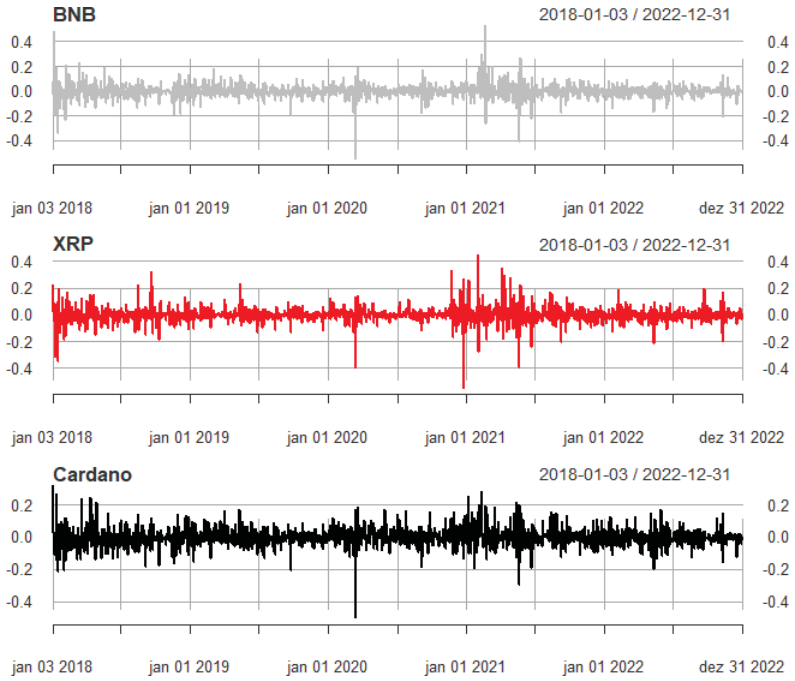
[3] Para facilitar a visualização, o trabalho seguirá com a separação de figuras e tabelas em 3 grupos, por ordem de liquidez: (i) O primeiro grupo unirá Bitcoin, Ethereum e Theter, (ii) o segundo será representado por BNB, XRP e Cardano e, (iii) o terceiro, terá Dogecoin, TRON, Litecoin e ChainLink.

GRÁFICO 2 – LOG-RETORNOS DAS CRIPTOMOEDAS (Grupo 1)



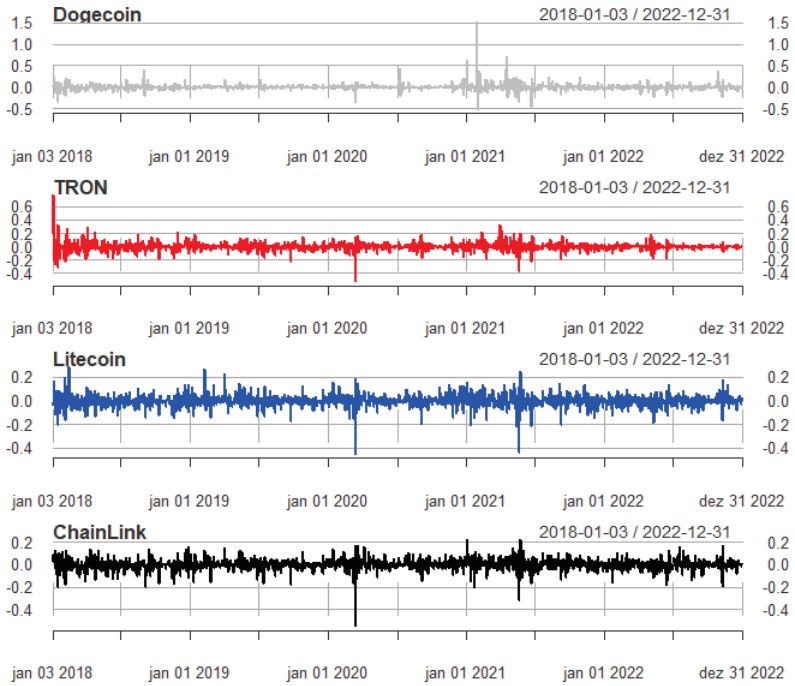
FONTE: CoinMarketCap (2023).

GRÁFICO 3 – LOG-RETORNOS DAS CRIPTOMOEDAS (Grupo 2)



FONTE: CoinMarketCap (2023).

GRÁFICO 4 – LOG-RETORNOS DAS CRIPTOMOEDAS (Grupo 3)



FONTE: CoinMarketCap (2023).

Os modelos utilizados nesta pesquisa consistem em um modelo Autoregressivo (ARIMA) para a média condicional e um tipo GARCH de primeira

ordem para a variância condicional. Os diferentes modelos GARCH analisados são nomeadamente: GARCH, EGARCH, GJRGARCH, Power ARCH (APARCH), Component GARCH (CSGARCH) e IGARCH.

O modelo ótimo é escolhido de acordo com três critérios de informação, nomeadamente Akaike (AIC), Bayesian (BIC) e RMSE, onde (i) os dois primeiros consideram o quão bom é o ajuste do modelo aos valores observados na amostra e o número de parâmetros utilizado, premiando um melhor ajuste e penalizando um número maior de parâmetros e, (ii) o terceiro leva em consideração a diferença entre os resultados do modelo de previsão com os valores reais. O modelo selecionado é aquele com os critérios mínimos de AIC, BIC e RMSE, sendo os dois primeiros para a análise *in-sample* e o último para a análise *out-of-sample.*

Importante destacar que os modelos ARIMA e GARCH apresentados neste trabalho, são criados a partir de pacotes distintos no R. Nesse sentido, tanto o AIC quanto o BIC não devem ser comparados entre essas diferentes classes de modelos.

Os pacotes “forecast” (ARIMA) e rugarch (GARCH) usam convenções distintas. Enquanto o “forecast” relata valores AIC/BIC com base em todas as observações diretamente, o “rugarch” os divide pelo número total de observações.

4 APRESENTAÇÃO DOS RESULTADOS

O presente capítulo tem por objetivo indicar os resultados encontrados no ensaio e se divide da seguinte maneira: (i) o item 4.1 apontará as estatísticas descritivas e os testes realizados na amostra. Nesse item também será explicado o processo para se chegar ao modelo ARIMA utilizado para cada criptomoeda; (ii) o item 4.2 tem por objetivo sinalizar o modelo GARCH, dado o modelo ARIMA previamente estabelecido, que tem um maior fit para o modelo *in-sample* e; (iii) o Item 4.3 busca um objetivo similar ao item anterior, se diferenciando na medida em que tem por objetivo analisar o comportamento dos modelos para a amostra *out-of-sample*.

4.1 ESTATÍSTICAS DESCRITIVAS E TESTES

As tabelas 1 a 3 apresentam as estatísticas resumidas para o fechamento diário dos log-retornos dos dez criptoativos. Como podemos observar na tabela 1: (i) o retorno médio diário do Bitcoin é igual a 0.005% com desvio padrão de 0,039; (ii) o Ethereum tem retorno médio diário igual a 0.016% com desvio padrão de 0,051; (iii) O Theter tem retorno médio diário igual a -0.0002% com desvio padrão de 0,004.

Tabela 1 – ESTATÍSTICAS DESCRITIVAS (Grupo 1)

Estatística	Bitcoin	Ethereum	Theter[4]
# Observações	1824	1824	1824
Média	0.00005	0.00016	-0.000002
Mediana	0.00097	0.00073	-0.000005
Mín	-0.46473	-0.55073	-0.052569
Máx	0.17182	0.23069	0.053392
Desvio Padrão	0.03897	0.05078	0.003930
Assimetria	-1.0766	-1.00303	0.329617
Curtose	13.25740	9.96718	42.621096

FONTE: CoinMarketCap (2023).

Em relação a tabela 2, podemos verificar: (i) o retorno médio diário do BNB é igual a 0.182% com desvio padrão de 0,057; (ii) o XRP tem retorno médio diário igual a -0.108% com desvio padrão de 0,059; (iii) O Cardano tem retorno médio diário igual a -0.063% com desvio padrão de 0,058.

[4] Para o Theter, foi incluída uma casa decimal adicional.

Tabela 2 - ESTATÍSTICAS DESCRITIVAS (Grupo 2)

Estatística	BNB	XRP	Cardano
# Observações	1824	1824	1824
Média	0.00182	-0.00108	-0.00063
Mediana	0.00064	-0.00123	-0.00036
Mín	-0.54308	-0.55050	-0.50365
Máx	0.52921	0.44475	0.32179
Desvio Padrão	0.05702	0.05945	0.05889
Assimetria	0.26402	-0.08410	-0.04936
Curtose	16.51083	12.76419	5.36762

FONTE: CoinMarketCap (2023).

Por fim, a tabela 3 apresenta os seguintes dados: (i) o retorno médio diário do Dogecoin é igual a 0.111% com desvio padrão de 0,076; (ii) o TRON tem retorno médio diário igual a -0.020% com desvio padrão de 0,061; (iii) O Litecoin tem retorno médio diário igual a -0.071% com desvio padrão de 0,054; (iv) O ChainLink tem retorno médio diário igual a 0.016% com desvio padrão de 0,051.

Tabela 3 - ESTATÍSTICAS DESCRITIVAS (Grupo 3)

Estatística	Dogecoin	TRON	Litecoin	ChainLink
# Observações	1824	1824	1824	1824
Média	0.00111	-0.00020	-0.00071	0.00016
Mediana	-0.00133	0.00094	-0.00001	0.00073
Mín	-0.51511	-0.52319	-0.44906	-0.55073
Máx	1.51638	0.78666	0.29059	0.23069
Desvio Padrão	0.07694	0.06167	0.05310	0.05078
Assimetria	5.03709	0.72586	-0.58109	-1.00303
Curtose	92.46143	20.67556	7.60583	9.96718

FONTE: CoinMarketCap (2023).

A tabela 4 apresenta os testes estatísticos para normalidade, estacionariedade e heterocedasticidade. O valor da estatística Jarque-Bera (JB) indica se os dados possuem ou não uma distribuição normal.

Já o valor do teste ADF (Dickey-Fuller Aumentado), indica se devemos rejeitar a hipótese nula de raiz unitária para os retornos[5], ou seja, indicando se a séria é ou não estacionária.

Por fim, o valor do teste ARCH (5) com 5 lags, para heterocedasticidade condicional indica se existem efeitos ARCH nos resíduos dos modelos ARIMA

[5] O teste ADF foi realizado tanto para a série de preços quanto para a série de log-retornos. Aqui indicamos apenas o resultado para a séria diferenciada, porém é possível verificar o mesmo teste para a série não diferenciada no arquivo script do R no Apêndice.

escolhidos, sugerindo que este modelo precisa ser expandido para incluir um modelo GARCH de ajuste de heteroscedasticidade condicional para a variância.

Tabela 4 - TESTES ESTATÍSTICOS

Cripto	Jarque-Bera	ADF	ARCH (5)
Bitcoin	13.747***	-45***	14***
Ethereum	7.878***	-45***	34***
Theter	138.416***	-67***	166***
BNB	20.793***	-44***	177***
XRP	12.418***	-43***	124***
Cardano	2.198***	-45***	61***
Dogecoin	658.932***	-42***	27***
TRON	32.731***	-45***	80***
Litecoin	4.513***	-45***	49***
ChainLink	7.878***	-45***	34***

FONTE: CoinMarketCap (2023).

*** Indica rejeição da hipótese nula ao nível de 1%

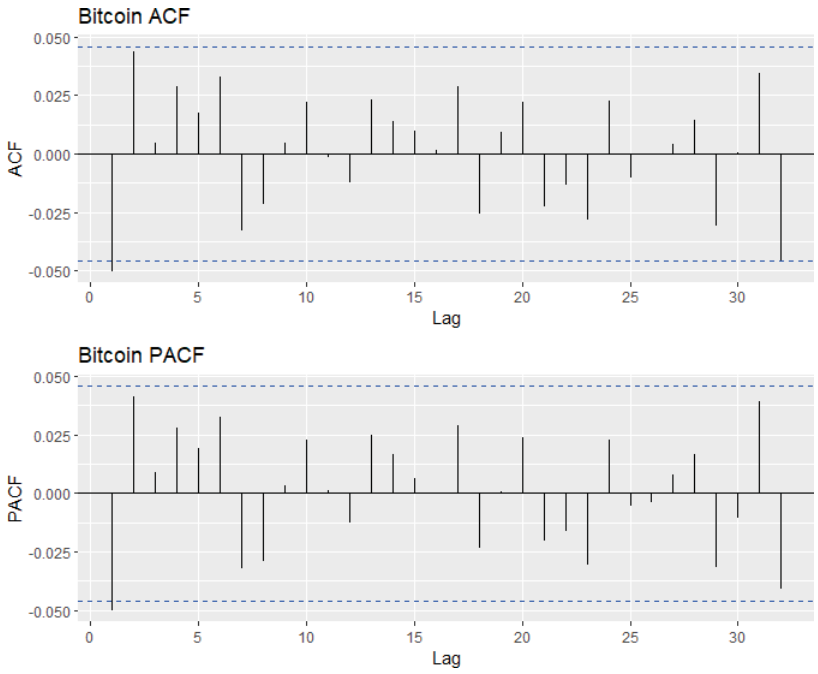
** Indica rejeição da hipótese nula ao nível de 5%

* Indica rejeição da hipótese nula ao nível de 10%

Os Gráficos 5 a 14 indicam as funções de autocorrelação e autocorrelação parcial[6] de cada uma das séries de log-retornos de cada um dos criptoativos, que foram usadas em conjunto com a função “auto.arima” do R e comparativos de AIC e BIC, para definição do modelo ARIMA que será implementado, posteriormente, nos modelos GARCH.

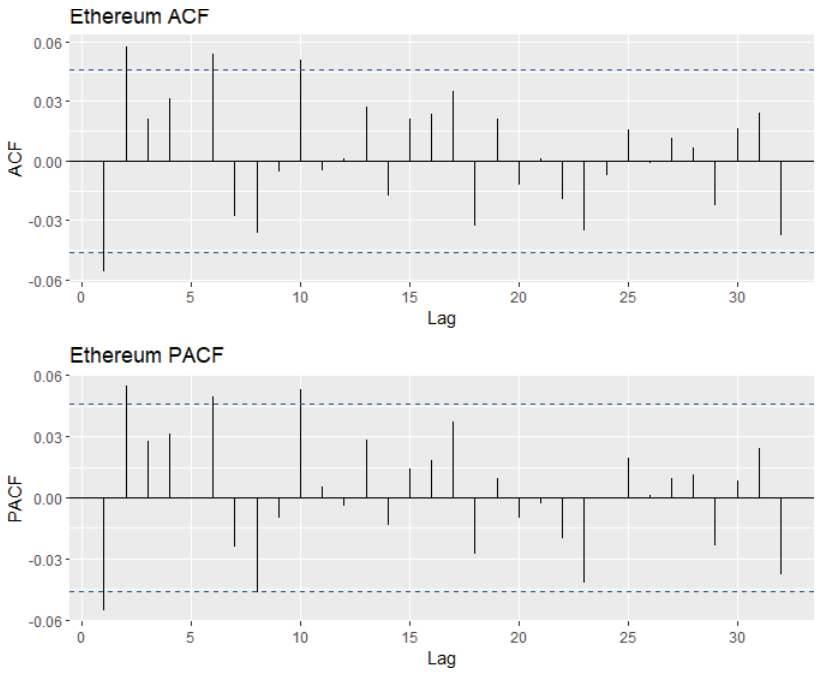
[6] A autocorrelação e a autocorrelação parcial são medidas de associação entre valores de séries atuais e anteriores e indicam quais valores de série anteriores são mais úteis para prever valores futuros.

GRÁFICO 5 – BITCOIN ACF E PACF



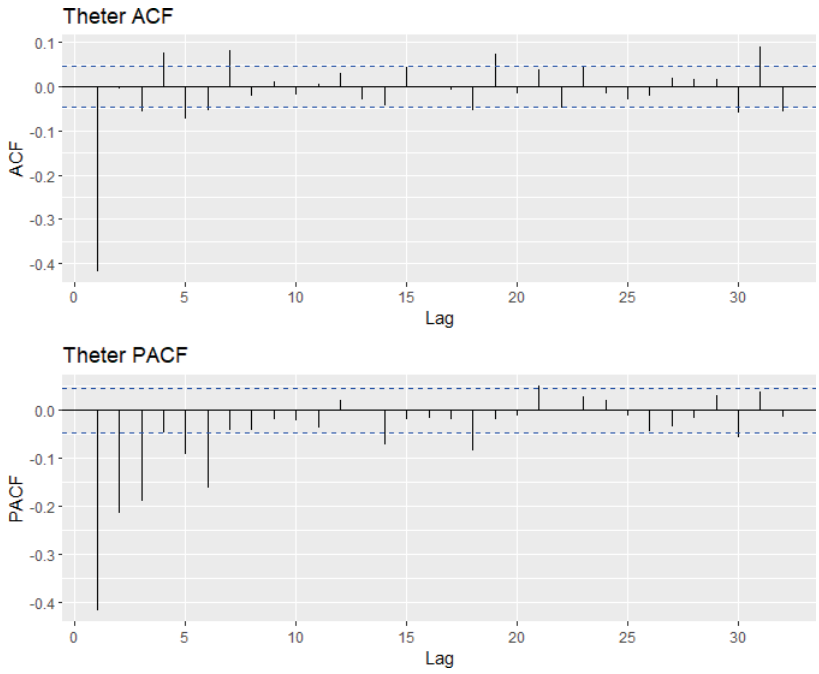
FONTE: CoinMarketCap (2023).

GRÁFICO 6 – ETHEREUM ACF E PACF



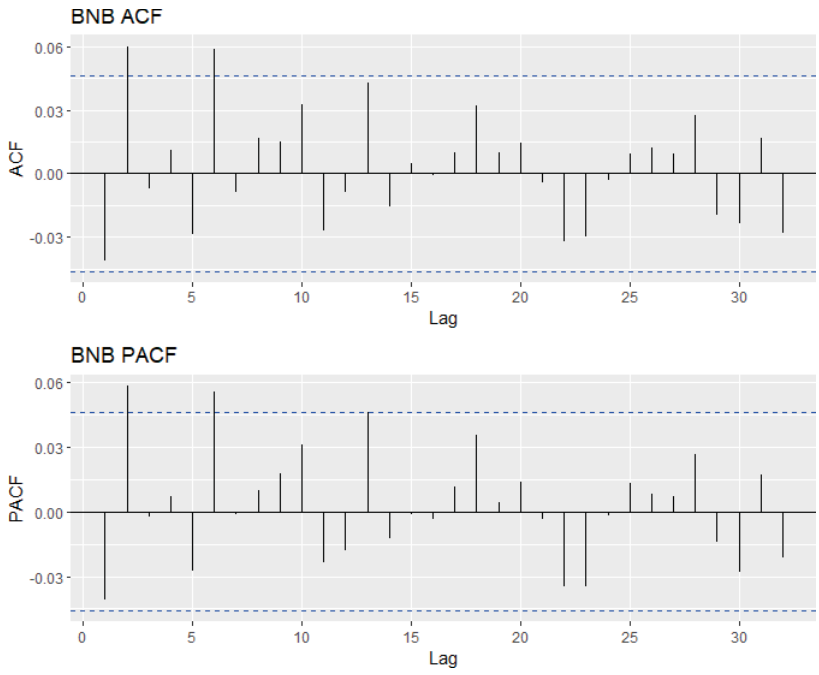
FONTE: CoinMarketCap (2023).

GRÁFICO 7 – THETER ACF E PACF



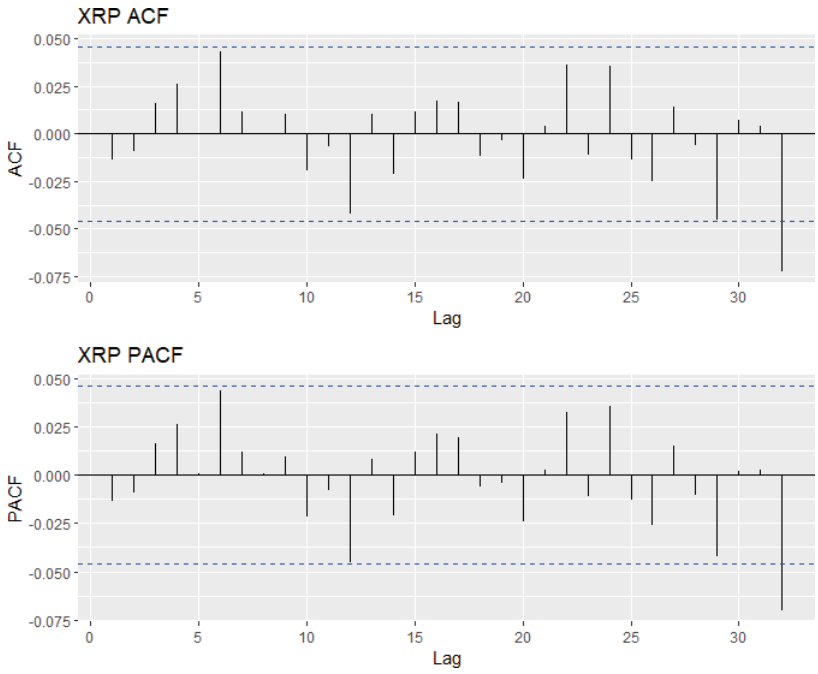
FONTE: CoinMarketCap (2023).

GRÁFICO 8 – BNB ACF E PACF



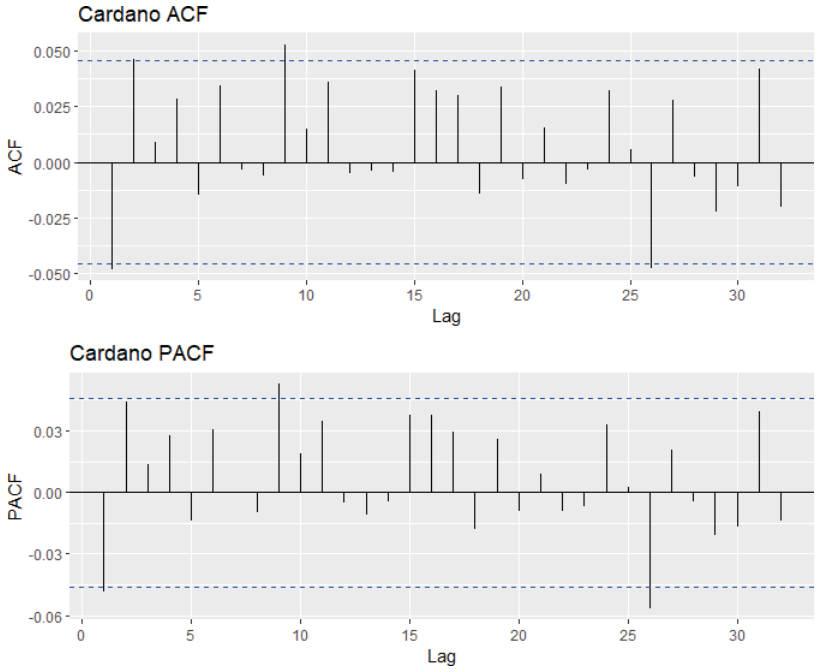
FONTE: CoinMarketCap (2023).

GRÁFICO 9 – XRP ACF E PACF



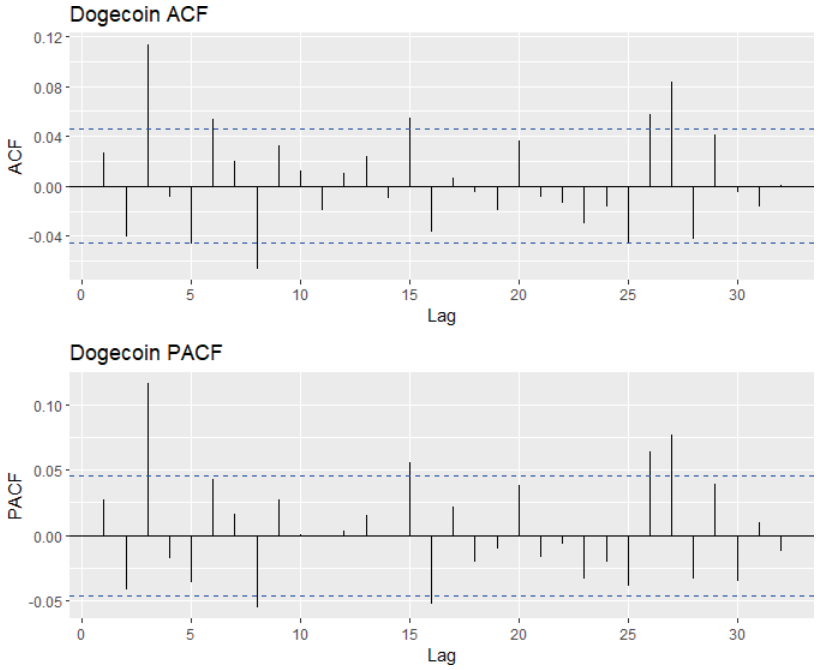
FONTE: CoinMarketCap (2023).

GRÁFICO 10 – CARDANO ACF E PACF



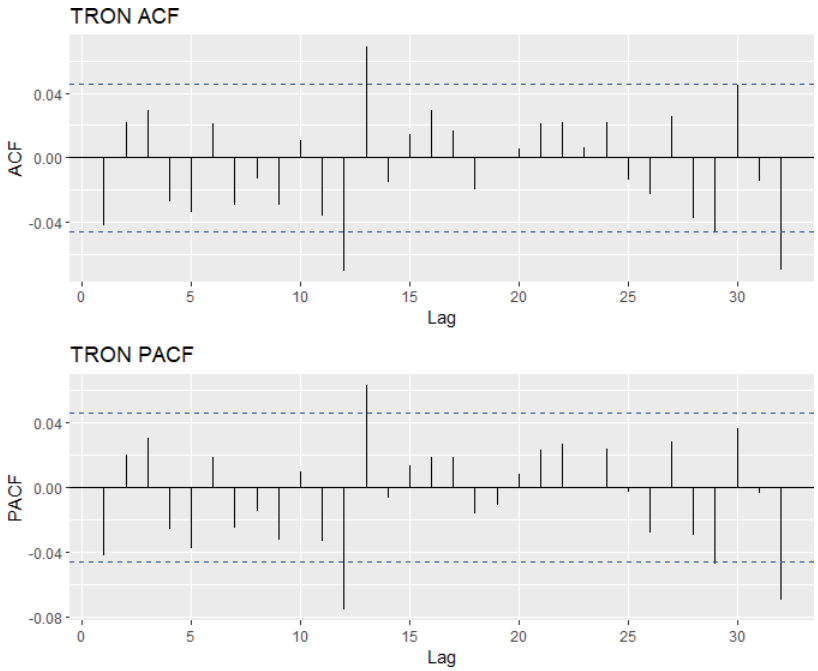
FONTE: CoinMarketCap (2023).

GRÁFICO 11 – DOGECOIN ACF E PACF



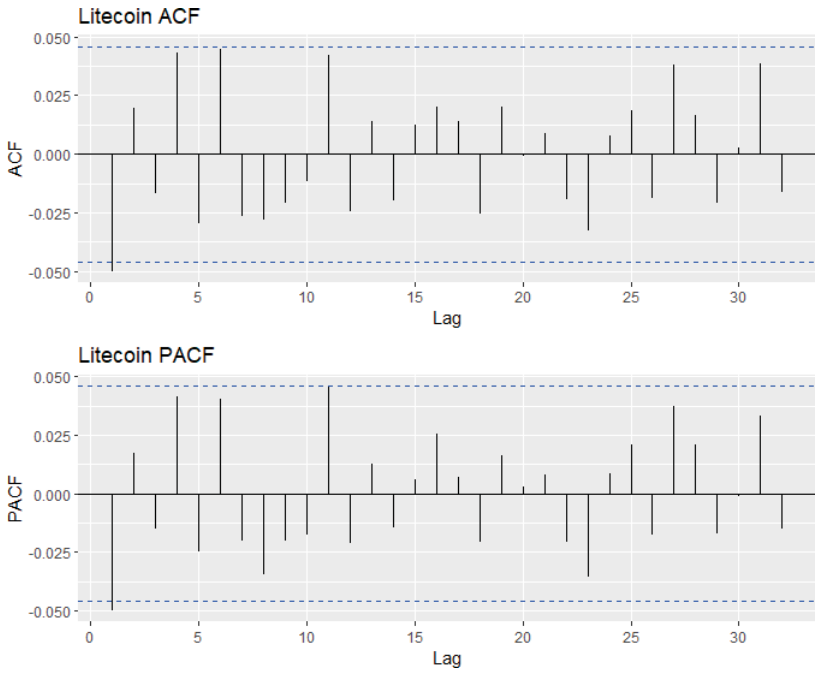
FONTE: CoinMarketCap (2023).

GRÁFICO 12 – TRON ACF E PACF



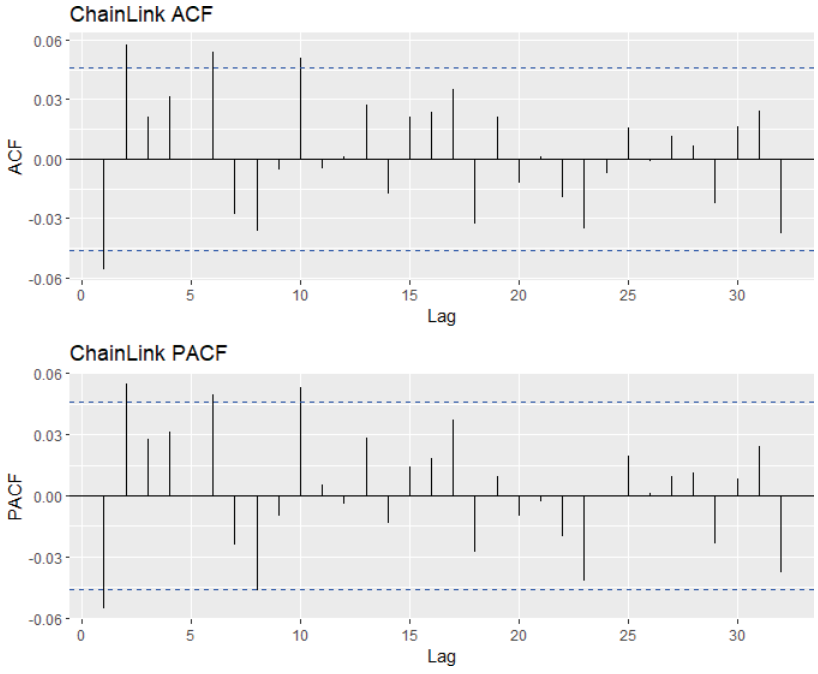
FONTE: CoinMarketCap (2023).

GRÁFICO 13 – LITECOIN ACF E PACF



FONTE: CoinMarketCap (2023).

GRÁFICO 14 – CHAINLINK ACF E PACF



FONTE: CoinMarketCap (2023).

Assim, levando em consideração os gráficos acima e, a função *auto.arima* do R, foram escolhidos três modelos para comparação de AIC e BIC, sendo o terceiro

modelo um intermediário entre o gráfico e a função *auto.arima*. A tabela 5 indica os modelos ARIMA que foram comparados e apresenta, em negrito, o modelo escolhido:

Tabela 5 – MODELOS ARIMA

Criptomoeda	ARIMA Gráfico	Auto Arima	Arima Intermediário
Bitcoin	(1,0,1)	(3,0,0)	**(1,0,0)**
Ethereum	(2,0,2)	(1,0,2)	**(1,0,1)**
Theter	(3,0,1)	**(5,0,5)**	(3,0,3)
BNB	**(0,0,0)**	(4,0,3)	(1,0,1)
XRP	(**0,0,0)**	(3,0,1)	(1,0,1)
Cardano	(1,0,2)	(1,0,2)	**(1,0,1)**
Dogecoin	(0,0,0)	**(1,0,1)**	(1,0,0)
TRON	**(0,0,0)**	(5,0,4)	(1,0,1)
Litecoin	**(1,0,1)**	**(1,0,1)**	(1,0,0)
ChainLink	(2,0,2)	(1,0,2)	**(1,0,1)**

FONTE: CoinMarketCap (2023).

Na tabela 6, é indicado de forma sumarizada o modelo escolhido para cada ativo. Importante destacar que o critério de escolha se deu com base no AIC e no BIC e, no caso de modelos com resultado similar, foi escolhido aquele que apresentava o menor número de parâmetros.

Tabela 6 – MODELO ARIMA ESCOLHIDO

Criptomoeda	ARIMA Escolhido
Bitcoin	(1,0,0)
Ethereum	(1,0,1)
Theter	(5,0,5)
BNB	(0,0,0)
XRP	(0,0,0)
Cardano	(1,0,1)
Dogecoin	(1,0,1)
TRON	(0,0,0)
Litecoin	(1,0,1)
ChainLink	(1,0,1)

FONTE: CoinMarketCap (2023).

4.2 RESULTADOS IN-SAMPLE

As tabelas 7 a 8 apresentam os resultados dos dois critérios de informação (AIC e BIC) para os diversos modelos GARCH analisados.

Tabela 7 – AIC

Cripto	ARIMA[7]	GARCH	EGARCH	GJRGARCH	APARCH	CSGARCH	IGARCH
Bitcoin	(1,0,0)	-3.7290	**-3.7408**	-3.7402	-3.7396	-3.7290	-3.7112
Ethereum	(1,0,1)	-3.1986	-3.1974	**-3.2016**	-3.2012	-3.1974	-3.1802
Theter	(5,0,5)	-10.104	**-10.246**	-10.138	-10.100	-10.060	-10.082
BNB	(0,0,0)	-3.1704	-3.1765	-3.1718	-3.1848	**-3.2032**	-3.1693
XRP	(0,0,0)	-3.0967	-3.1040	-3.0956	-3.0960	**-3.1402**	-3.0974
Cardano	(1,0,1)	-2.9276	-2.9337	-2.9281	-2.9270	**-2.9380**	-2.9128
Dogecoin	(1,0,1)	-2.7862	-2.8867	-2.8362	-2.8528	**-2.9432**	-2.7883
TRON	(0,0,0)	-3.0698	-3.0691	-3.0694	-3.0713	**-3.1095**	-3.0710
Litecoin	(1,0,1)	-3.1104	**-3.1124**	-3.1104	-3.1101	-3.1105	-3.0917
ChainLink	(1,0,1)	-3.1986	-3.1974	**-3.2016**	-3.2012	-3.1974	-3.1802

FONTE: CoinMarketCap (2023).

Tabela 8 – BIC

Cripto	ARIMA[4]	GARCH	EGARCH	GJRGARCH	APARCH	CSGARCH	IGARCH
Bitcoin	(1,0,0)	-3.7139	**-3.7227**	-3.7221	-3.7185	-3.7079	-3.6991
Ethereum	(1,0,1)	**-3.1805**	-3.1763	**-3.1805**	-3.1770	-3.1732	-3.1651
Theter	(5,0,5)	-10.062	**-10.201**	-10.093	-10.052	-10.012	-10.042
BNB	(0,0,0)	-3.1583	-3.1614	-3.1567	-3.1667	**-3.1851**	-3.1603
XRP	(0,0,0)	-3.0846	-3.0889	-3.0805	-3.0779	**-3.1220**	-3.0884
Cardano	(1,0,1)	-2.9095	-2.9125	-2.9069	-2.9028	**-2.9139**	-2.8977
Dogecoin	(1,0,1)	-2.7681	-2.8655	-2.8150	-2.8287	**-2.9191**	-2.7732
TRON	(0,0,0)	-3.0577	-3.0540	-3.0543	-3.0532	**-3.0914**	-3.0619
Litecoin	(1,0,1)	**-3.0923**	-3.0913	-3.0892	-3.0860	-3.0864	-3.0766
ChainLink	(1,0,1)	**-3.1805**	-3.1763	**-3.1805**	-3.1770	-3.1732	-3.1651

FONTE: CoinMarketCap (2023).

Para o Bitcoin, o modelo que possui os melhores critérios de informação é o modelo AR-EGARCH, com AIC de -3,7408 e BIC de -3,7227.

Em relação ao Ethereum, o modelo que possui os melhores valores de critério de informação é o modelo AR-GJRGARCH, com AIC de -3,2016 e BIC de -3,1805. Importante destacar que para esse segundo criptoativo, o modelo GARCH, apresenta BIC que iguala o modelo GJRGARCH.

O modelo que possui os melhores critérios de informação para o Theter, é o modelo EGARCH, similar ao que ocorreu ao Bitcoin, com AIC -10.246 e BIC -10.201.

Em relação ao BNB, o modelo que possui os melhores valores de critério de informação é o modelo CSGARCH, com AIC de -3,2032 e BIC de -3,1851.

Para o XRP, o modelo que possui os melhores critérios de informação é o modelo CSGARCH, com AIC de -3,1402 e BIC de -3,1220.

---

[7] Modelo ARIMA usado para cada criptomoeda com base na comparação entre ACF, PACF e função "auto.arima" do R. Modelo escolhido foi replicado para todos os modelos GARCH.

Já o Cardano se comportou melhor também com o modelo CSGARCH, tendo AIC de -2,9380 e BIC de 2,9139.

No que diz respeito a Dogecoin e TRON, também foram observados resultados superiores através do modelo CSGARCH. Nesse caso, foi obtido um AIC de -2,9432 e um BIC de -2,9191 para Dogecoin e AIC de -3,1095 e BIC de -3,0914 para TRON.

Por fim, Litecoin e ChainLink tiveram resultados discutíveis. No caso de Litecoin, EGARCH e GARCH tiveram AIC e BIC superiores, respectivamente. Já para ChainLink, o resultado indica que o melhor modelo é o GJRGARCH. Entretanto, para o BIC, o GARCH se comportou de forma equivalente.

4.3 RESULTADOS OUT-OF-SAMPLE

A tabela 9 apresenta os resultados do RMSE para os diversos modelos GARCH analisados na amostra *out-of-sample*. Importante destacar que para cada modelo foi comparado o resultado da sua volatilidade prevista com a volatilidade realizada nos primeiros 30 dias após o fim da amostra, ou seja, os primeiros 30 dias de 2023.

Tabela 9 – RMSE

Cripto	ARIMA[8]	GARCH	EGARCH	GJRGARCH	APARCH	CSGARCH	IGARCH
Bitcoin	(1,0,0)	0,01355	0,01404	**0,01091**	0,01193	0,01374	0,01849
Ethereum	(1,0,1)	0,02254	**0,01386**	0,01980	0,01858	0,02136	0,01998
Theter	(5,0,5)	0,00011	0,00049	0,00016	0,00014	0,00016	**0,00010**
BNB	(0,0,0)	0,02232	**0,01656**	0,02163	0,02319	0,02166	0,02179
XRP	(0,0,0)	0,04287	0,03265	0,02705	0,04009	**0,01246**	0,03528
Cardano	(1,0,1)	0,01819	0,01613	0,01834	0,01885	**0,01364**	0,01949
Dogecoin	(1,0,1)	0,02295	0,03584	0,01877	**0,01365**	0,03826	0,02152
TRON	(0,0,0)	0,02235	0,01755	0,02150	0,02094	**0,00925**	0,01993
Litecoin	(1,0,1)	0,01715	0,01692	0,01935	0,01944	**0,01375**	0,01567
ChainLink	(1,0,1)	0,02254	**0,01386**	0,01980	0,01858	0,02136	0,01998

FONTE: CoinMarketCap (2023).

[8] Modelo ARIMA usado para cada criptomoeda com base na comparação entre ACF, PACF e função "auto.arima" do R. Modelo escolhido foi replicado para todos os modelos GARCH.

Para o Bitcoin, o modelo que possui o melhor critério de informação é o modelo AR-GJRGARCH, com RMSE de 0,01091.

Em relação ao Ethereum, o modelo que possui o melhor valor de critério de informação é o modelo AR-EGARCH, com RMSE de 0,01386. Resultado similar encontrato para os ativos BNB, onde esse modelo teve RMSE de 0,01656 e para o Chainlink, com RMSE de 0,01386.

O modelo que possui o melhor resultado para o Theter, é o modelo AR-IGARCH, com RMSE de 0,00010.

Para o XRP, o modelo que possui o melhor critério de informação é o modelo AR-CSGARCH, com RMSE de 0,01246. O mesmo ocorre para Cardano, TRON e Litecoin, com RMSE, respectivamente de 0,01364, 0,00925 e 0,01375.

Por fim, Dogecoin teve uma melhor performance com o modelo APARCH, com RMSE de 0,01365.

5 CONSIDERAÇÕES FINAIS

O resultado encontrado para os criptoativos estudados é especialmente interessante, pois um dos problemas do modelo GARCH convencional é que ele assume que choques positivos e negativos têm os mesmos efeitos na volatilidade, visto que este depende do quadrado dos choques anteriores, conforme descrito em Tsay (2010).

Sabemos, entretanto, que boa parte dos ativos financeiros, e aqui podemos incluir as criptomoedas, respondem de forma diferente a choques positivos e negativos. Além desse efeito assimétrico dos choques, outro ponto de especial desses ativos é o componente transitório da volatilidade.

Nesse sentido, os resultados encontrados nesse estudo são particularmente interessantes para o universo dos criptoativos, pois este é um mercado relativamente especulativo, onde analistas recorrentemente indicam uma possível existência de bolhas. Assim, o resultado nos mostra que modelos GARCH que: (i) incorporam efeitos assimétricos na volatilidade (como EGARCH e GJRGARCH), tendem a explicar melhor o comportamento das criptomoedas mais maduras e; (ii) ativos menos maduros e líquidos, ou seja, mais suscetíveis a especulação, tendem a ser mais bem explicados por modelos que decompõem a variância condicional em dois componentes: um componente permanente e outro transitório, de modo a investigar os componentes de longo e curto prazo da volatilidade da série do ativo (como o CSGARCH).

De forma resumida, Bitcoin, Ethereum e ChainLink tiveram um comportamento similar tanto na análise *in-sample* quanto *out-of-sample*, tendo os modelos EGARCH e GJRGARCH performado melhor que os outros. O Theter também entraria nesse grupo, porém o resultado *out-of-sample* indicou que os modelos assimétricos não tiveram um bom fit com o ativo.

Já para as outras moedas, o modelo CSGARCH foi aquele que se comportou melhor no geral, para o BNB o modelo foi o que teve o melhor fit *in-sample* e o segundo melhor *out-of-sample* (com o modelo GJRGARCH também se comportando bem). Já para o Litecoin teve uma performance melhor *out-of-sample*.

Por fim, as criptomoedas XRP, Cardano, Dogecoin Etron tiveram uma superioridade indiscutível para o modelo CSGARCH, tanto *in-sample* quanto *out-of-sample*.

Nosso estudo diverge de Chu et al. (2017) em relação aos modelos GARCH mais adequados para certas criptomoedas. Embora seu trabalho tenha identificado o IGARCH e GJRGARCH como os mais adequados para moedas estabelecidas como Bitcoin e Litecoin, nossos resultados sugerem uma gama mais ampla de modelos dependendo da maturidade e liquidez da criptomoeda. Para criptomoedas mais estabelecidas e líquidas, modelos como EGARCH e GJRGARCH, que capturam efeitos de volatilidade assimétrica, tiveram melhor desempenho em nosso estudo. Isso se alinha com as descobertas de Chu et al. para moedas maiores. No entanto, para moedas menos maduras e líquidas, descobrimos que o modelo CSGARCH, que decompõe a volatilidade em componentes permanentes e transitórios, é o modelo mais adequado. Isso sugere que essas moedas podem experimentar maiores flutuações de volatilidade de curto prazo, destacando a importância de considerar as características específicas de cada criptomoeda ao selecionar um modelo GARCH.

Em relação ao trabalho de Katsiampa (2017), os resultados deste estudo divergem em alguns aspectos. Enquanto esse estudo indicou o CSGARCH como o modelo mais adequado para o Bitcoin, nossos resultados sugerem que modelos como EGARCH e GJRGARCH podem ser mais adequados para algumas das criptomoedas analisadas, incluindo o próprio Bitcoin (moeda analisa pelo trabalho em questão). Essa diferença de resultado é especialmente interessante na medida em que o Bitcoin naquele momento ainda possuía uma liquidez relativamente baixa em relação ao período atual. Nesse contexto, a evolução do mercado de criptomoedas nos últimos anos pode ter alterado as características de volatilidade desses ativos e, ainda que mais dados sejam requeridos para que se possa afirmar, o modelo CSGARCH se comporta bem para moedas menos liquidas e maduras, o que pode ter sido o caso do Bitcoin naquele momento, onde o efeito de curto prazo da volatilidade era o fator preponderante para a variância do ativo.

Esse trabalho teve como objetivo geral, identificar o modelo GARCH que melhor se aplica para cada um dos criptoativos estudados, tanto *in-sample* quanto *out-of-sample*. Nesse sentido, o estudo permite abrir a discussão sobre o tema e possui diversas possibilidades de prosseguimento do ponto de vista de objeto de estudo, entre eles podem ser citados: (i) replicar o estudo com a inclusão de diferentes distribuições de probabilidade; (ii) ampliar o escopo para uma carteira de criptoativos, onde o pesquisador poderia abordar melhor o efeito da volatilidade dessas moedas em um ambiente de gestão de risco de carteira, em que a correlação entre os ativos

pode exercer um papel importante na definição do modelo GARCH adequado e; (iii) sabendo-se do resultado encontrado para criptomoedas com menor liquidez, utilizar a mesma metodologia para criptomoedas situadas abaixo destas no ranking do CoinMarketCap, com o intuito de avaliar se os efeitos transitórios do CSGARCH também explicam de forma satisfatória o comportamento dessas moedas menores.

REFERÊNCIAS

AKAIKE, H. A new look at the statistical model identification. **IEEE Transactions on Automatic Control**, [*s. l.*], v. 19, p. 716-723, 1974.

BASHAR, A. ; MUNEER, A; AMMAR, A. (2021). Performance of ARCH and GARCH Models in Forecasting Cryptocurrency Market Volatility. **Industrial Engineering & Management Systems**. 20. 130-139. 10.7232/iems.2021.20.2.130.

ASTERIOU, D.; HALL, S. **Applied Econometrics**. 4ª edição. ed. [*S. l.*]: Bloomsbury Publishing, 2021. 568 p. ISBN 9781352012026.

BERGSLI, L.O.; LIND, A.F.; MOLNÁR, P.; POLASIK, M. Forecasting volatility of Bitcoin. **Research in International Business and Finance**, [*s. l.*], v. 59, 2022.

BITCOIN WHITE PAPER. Disponível em: https://bitcoin.org/en/bitcoin-paper. Acesso em: 1 jan. 2024.

BOLLERSLEV, T. Generalized autoregressive conditional heteroskedasticity. **Journal of Econometrics**, [*s. l.*], v. 31, p. 307-327, 1986.

CHU, J.; CHAN, S.; NADARAJAH, S.; OSTERRIEDER, J. GARCH Modelling of Cryptocurrencies. **Journal of Risk and Financial Management**, [*s. l.*], v. 10, p. 17-17, 2017.

CRYPTO-CURRENCY market capitalizations. CoinMarketCap, 2022. Disponível em: https://coinmarketcap.com/. Acesso em: 30 jan. 2024.

ENOKSEN, F.A; LANDSNES, J.CH; LU?IVJANSKÁ, K.; MOLNÁR, P. Understanding risk of bubbles in cryptocurrencies. **Journal of Economic Behavior & Organization**, [*s. l.*], v. 176, p. 129-144, 2020.

ETHEREUM WHITE PAPER. Disponível em: https://ethereum.org/pt/what-is-ethereum/. Acesso em: 1 jan. 2024.

GHALANOS, A. **Univariate GARCH models**. 1.4. [*S. l.*], 2018. R package version 1.4. Disponível em: https://cran.r-project.org/web/packages/rugarch/rugarch.pdf. Acesso em: 1 jan. 2023.

IMF CRYPTO DATA. Disponível em: https://www.imf.org/en/Topics/fintech/central-bank-digital-currency/virtual-handbook. Acesso em: 1 jan. 2024.

KATSIAMPA, P. Volatility estimation for Bitcoin: A comparison of GARCH models. **Economics Letters**, [*s. l.*], v. 158, p. 3-6, 2017.

KOTHARI, S.P.; ZIMMERMAN, J.L. Price and return models,. **Journal of Accounting and Economics**, [*s. l.*], v. 20, p. 155-192, 1995.

KYEI-MENSAH, J. Asymmetric Reverting and Volatility in Cryptocurrency Markets. Available at SSRN: https://ssrn.com/abstract=4844144 or http://dx.doi.org/10.2139/ssrn.4844144

MORETTIN, P.A. **Econometria Financeira**: um curso em séries temporais financeiras. 3ª edição. ed. [*S. l.*]: Blucher, 2016. 420 p. ISBN 8521211309.

TSAY, R.S. **Analysis of financial time series**. 3ª edição. ed. atual. [*S. l.*]: JOHN WILEY & SONS, 1951. ISBN 978-0-470-41435-4.

APÊNDICE – CÓDIGO DO TRABALHO NO R

Apresenta-se, neste apêndice, o código desenvolvido pelo autor. O código foi desenvolvido para ser executado no software livre R (versão 4.3.1), disponível em: https://cran.r-project.org/bin/windows/base/.

Código:

```
##### Dissertação #####

##Import Packages##

## If it's the first time running the code, use the install.packages()
function for each library.

library(tidyverse)
library(xts)
library(lubridate)
library(stringr)
library(tseries)
library(forecast)
library(GetBCBData)
library(lmtest)
library(FinTS)
library(rugarch)
library(fGarch)
library(devtools)
library(dynlm)
library(tsm)
library(remotes)
#remotes::install_github("KevinKotze/tsm") # Only remove the "#" if it's
the first time running the code
#devtools::install_github("sstoeckl/crypto2") # Only remove the "#" if it's
the first time running the code.
library(crypto2)
library(flexmix)
library(nortsTest)
library(scales)
library(gridExtra)
library(psych)
library(pastecs)
library(ggplot2)
library(bayesforecast)
library(dplyr)

options(scipen=999,"digits"=2)


#### Getting the list of cryptos ####
coins = crypto_list(only_active = TRUE)

##filtering to get the top 15 coins, with data between jan/18 and dec/22,
by coin market cap rank ##
coins = filter(coins, first_historical_data <= "2018-01-01")
coins = filter(coins, last_historical_data >= "2022-12-31")
coins = arrange(coins, rank)
coins = coins[0:10,]
```

```
## Getting historical values ##
crypto = crypto_history(coin_list = coins, convert = "USD", limit = NULL,
start_date = "20180101", end_date = "20221231", interval = NULL, sleep =
60, finalWait = TRUE)

## Selecting the determined columns
crypto = select(crypto,c("timestamp","name","symbol","close"))

## Adjusting Values
crypto$close = as.numeric(str_replace(crypto$close, ',', '.'))
crypto$timestamp = as.Date(crypto$timestamp)
crypto <- crypto %>%
  mutate(name = ifelse(name == "Tether USDt", "Tether", name))

coins <- coins %>%
  mutate(name = ifelse(name == "Tether USDt", "Tether", name))

## Getting a DataFrame for each Crypto ##
i = 1

for (i in 1:10) {

  assign(paste(coins$name[i],"_df",sep = ""), filter(crypto, crypto$name ==
coins$name[i]))


}

#### Creating time series for each Crypto and Getting the statistics ####

bitcoin_ts <- xts(Bitcoin_df$close, order.by=Bitcoin_df$timestamp)
ethereum_ts <- xts(Ethereum_df$close, order.by = Ethereum_df$timestamp)
tether_ts <- xts(Tether_df$close, order.by = Tether_df$timestamp)
BNB_ts <- xts(BNB_df$close, order.by = BNB_df$timestamp)
XRP_ts <- xts(XRP_df$close, order.by = XRP_df$timestamp)
cardano_ts <- xts(Cardano_df$close, order.by = Cardano_df$timestamp)
dogecoin_ts <- xts(Dogecoin_df$close, order.by = Dogecoin_df$timestamp)
TRON_ts <- xts(TRON_df$close, order.by = TRON_df$timestamp)
litecoin_ts <- xts(Litecoin_df$close, order.by = Litecoin_df$timestamp)
chainlink_ts <- xts(Ethereum_df$close, order.by = Ethereum_df$timestamp)

### Plotting and getting the statistics + Stationarity test (ADF) ##

xts_list =
merge(bitcoin_ts,ethereum_ts,tether_ts,BNB_ts,XRP_ts,cardano_ts,dogecoin_ts
,TRON_ts,litecoin_ts,chainlink_ts)

i = 1
j = ncol(xts_list)
for (i in 1:j) {

  print(plot.xts(xts_list[,i],main = colnames(xts_list[,i])))
  assign(paste("summary_",colnames(xts_list[,i]),sep =
""),summary(xts_list[,i]))
  assign(paste("adf_",colnames(xts_list[,i]),sep =
""),adf.test(xts_list[,i], alternative = "stationary", k=0)) # p-value <
0.05 indicates the TS is stationary

}
```

```
### Listing the coins to make it easier to run the codes + Differentiating
them, to make it stationary ###
xts_list_Diff = diff(log(xts_list), lag = 1)
xts_list_Diff = xts_list_Diff[-1,]

### Running again the statistics and testing for stationarity in the diff
series ###

i = 1
j = ncol(xts_list_Diff)

for (i in 1:j) {

  print(plot.xts(xts_list_Diff[,i],main = colnames(xts_list_Diff[,i])))
  assign(paste("summary_diff_",colnames(xts_list_Diff[,i]),sep =
""),summary(xts_list_Diff[,i]))
  assign(paste("adf_diff_",colnames(xts_list_Diff[,i]),sep =
""),adf.test(xts_list_Diff[,i], alternative = "stationary", k=0)) # p-value
< 0.05 indicates the TS is stationary

}

## Jarque Bera ##

jarque.bera.test(xts_list_Diff$bitcoin_ts)
jarque.bera.test(xts_list_Diff$ethereum_ts)
jarque.bera.test(xts_list_Diff$tether_ts)
jarque.bera.test(xts_list_Diff$BNB_ts)
jarque.bera.test(xts_list_Diff$XRP_ts)
jarque.bera.test(xts_list_Diff$cardano_ts)
jarque.bera.test(xts_list_Diff$dogecoin_ts)
jarque.bera.test(xts_list_Diff$TRON_ts)
jarque.bera.test(xts_list_Diff$litecoin_ts)
jarque.bera.test(xts_list_Diff$chainlink_ts)


#### ARIMA Model ####

i = 1
j = ncol(xts_list_Diff)

for (i in 1:j) {

  print(ggAcf(xts_list_Diff[,i],main = colnames(xts_list_Diff[,i])))
  print(ggPacf(xts_list_Diff[,i],main = colnames(xts_list_Diff[,i])))
  assign(paste("ARIMA_",colnames(xts_list_Diff[,i]),sep =
""),auto.arima(xts_list_Diff[,i]))

}

### Choosing 3 possible ARIMA models for each Crypto ###
# The order will always be 1) auto.arima, 2)Graph decision, 3) An
intermediate option.
## In case auto.arima and graph is the same, another intermediate option
will be chosen.


#Bitcoin
ARIMA_bitcoin_ts #Graph (1,0,1) and auto.arima indicates (3,0,0)

bitcoin_model_ARIMA_1 = Arima(xts_list_Diff$bitcoin_ts, order=c(3,0,0))
```

```
bitcoin_model_ARIMA_2 = Arima(xts_list_Diff$bitcoin_ts, order=c(1,0,1))
bitcoin_model_ARIMA_3 = Arima(xts_list_Diff$bitcoin_ts, order=c(1,0,0))

AIC(bitcoin_model_ARIMA_1)
AIC(bitcoin_model_ARIMA_2)
AIC(bitcoin_model_ARIMA_3)

BIC(bitcoin_model_ARIMA_1)
BIC(bitcoin_model_ARIMA_2)
BIC(bitcoin_model_ARIMA_3)

## ARIMA (1,0,1) is the one who has the best AIC.
## ARIMA (1,0,0) is the one who has the best BIC.
# I'll choose the one with the better BIC, model 3. As the difference
between models is greater in the BIC comparison and it has lower
parameters.

#Ethereum
ARIMA_ethereum_ts #Graph (2,0,2) and auto.arima indicates (1,0,2)

ethereum_model_ARIMA_1 = Arima(xts_list_Diff$ethereum_ts, order=c(1,0,2))
ethereum_model_ARIMA_2 = Arima(xts_list_Diff$ethereum_ts, order=c(2,0,2))
ethereum_model_ARIMA_3 = Arima(xts_list_Diff$ethereum_ts, order=c(1,0,1))

AIC(ethereum_model_ARIMA_1)
AIC(ethereum_model_ARIMA_2)
AIC(ethereum_model_ARIMA_3)

BIC(ethereum_model_ARIMA_1)
BIC(ethereum_model_ARIMA_2)
BIC(ethereum_model_ARIMA_3)

## ARIMA (1,0,2) is the one who has the best AIC.
## ARIMA (1,0,1) is the one who has the best BIC.
# I'll choose the one with the better BIC, model 3. As the difference
between models is not big, I prefer to choose the one with lower
parameters.


#Theter
ARIMA_tether_ts # Graph indicates ARIMA (3,0,1) instead of (5,0,5), should
try to run AIC for some combinations

tether_model_ARIMA_1 = Arima(xts_list_Diff$tether_ts, order=c(5,0,5))
tether_model_ARIMA_2 = Arima(xts_list_Diff$tether_ts, order=c(3,0,1))
tether_model_ARIMA_3 = Arima(xts_list_Diff$tether_ts, order=c(3,0,3))

AIC(tether_model_ARIMA_1)
AIC(tether_model_ARIMA_2)
AIC(tether_model_ARIMA_3)

BIC(tether_model_ARIMA_1)
BIC(tether_model_ARIMA_2)
BIC(tether_model_ARIMA_3)

## ARIMA (5,0,5) is the one who has the best AIC.
## ARIMA (5,0,5) and (3,0,3) are the ones with the best BIC.
# I'll choose to continue with model 1, as it has better AIC and the same
BIC as model 3 (second best).

#BNB
```

```
ARIMA_BNB_ts #Graph indicates (0,0,0) instead of (4,0,3). Should try some
combinations but I'd go with the graph to reduce the numbers of variables

BNB_model_ARIMA_1 = Arima(xts_list_Diff$BNB_ts, order=c(4,0,3))
BNB_model_ARIMA_2 = Arima(xts_list_Diff$BNB_ts, order=c(0,0,0))
BNB_model_ARIMA_3 = Arima(xts_list_Diff$BNB_ts, order=c(1,0,1))

AIC(BNB_model_ARIMA_1)
AIC(BNB_model_ARIMA_2)
AIC(BNB_model_ARIMA_3)

BIC(BNB_model_ARIMA_1)
BIC(BNB_model_ARIMA_2)
BIC(BNB_model_ARIMA_3)

## ARIMA (4,0,3) is the one who has the best AIC.
## ARIMA (0,0,0) is the one with the best BIC.
# Although model 2 is not the best in both AIC and BIC, it is the second
best option in both comparisons and seems to be the best option to
continue.

#XRP
ARIMA_XRP_ts #Graph indicates (0,0,0), completely different from the
(3,0,1) of the auto.arima. Need to understand the reason for this
difference.

XRP_model_ARIMA_1 = Arima(xts_list_Diff$XRP_ts, order=c(3,0,1))
XRP_model_ARIMA_2 = Arima(xts_list_Diff$XRP_ts, order=c(0,0,0))
XRP_model_ARIMA_3 = Arima(xts_list_Diff$XRP_ts, order=c(1,0,1))

AIC(XRP_model_ARIMA_1)
AIC(XRP_model_ARIMA_2)
AIC(XRP_model_ARIMA_3)

BIC(XRP_model_ARIMA_1)
BIC(XRP_model_ARIMA_2)
BIC(XRP_model_ARIMA_3)

## ARIMA (0,0,0) is the one who has the best AIC.
## ARIMA (0,0,0) is the one with the best BIC.
# model 2 is the best in both AIC and BIC

#Cardano
ARIMA_cardano_ts #Graph indicates (1,0,2), same as the auto.arima.

Cardano_model_ARIMA_1 = Arima(xts_list_Diff$cardano_ts, order=c(1,0,2))
Cardano_model_ARIMA_2 = Arima(xts_list_Diff$cardano_ts, order=c(1,0,1))
Cardano_model_ARIMA_3 = Arima(xts_list_Diff$cardano_ts, order=c(2,0,2))

AIC(Cardano_model_ARIMA_1)
AIC(Cardano_model_ARIMA_2)
AIC(Cardano_model_ARIMA_3)

BIC(Cardano_model_ARIMA_1)
BIC(Cardano_model_ARIMA_2)
BIC(Cardano_model_ARIMA_3)

## ARIMA (2,0,2) is the one who has the best AIC. 1 point above the other
options.
## ARIMA (1,0,1) is the one with the best BIC.
```

```
# Altough model 1 seems to be the best accordingly to auto.arima and graph,
model 2 is the best in BIC and will be chosen.

#Dogecoin
ARIMA_dogecoin_ts #Graph indicates (0,0,0), distinct to the (1,0,1) of the
auto.arima function.

Dogecoin_model_ARIMA_1 = Arima(xts_list_Diff$dogecoin_ts, order=c(1,0,1))
Dogecoin_model_ARIMA_2 = Arima(xts_list_Diff$dogecoin_ts, order=c(0,0,0))
Dogecoin_model_ARIMA_3 = Arima(xts_list_Diff$dogecoin_ts, order=c(1,0,0))

AIC(Dogecoin_model_ARIMA_1)
AIC(Dogecoin_model_ARIMA_2)
AIC(Dogecoin_model_ARIMA_3)

BIC(Dogecoin_model_ARIMA_1)
BIC(Dogecoin_model_ARIMA_2)
BIC(Dogecoin_model_ARIMA_3)

## ARIMA (1,0,1) is the one who has the best AIC.
## ARIMA (0,0,0) is the one with the best BIC.
# Model 1 will be chosen as it has the best AIC and its quite close model 2
in the BIC.

#Tron
ARIMA_TRON_ts #Graph indicates (0,0,0), quite different from the (5,0,4) of
the auto.arima. Should try to understand better.

Tron_model_ARIMA_1 = Arima(xts_list_Diff$TRON_ts, order=c(5,0,4))
Tron_model_ARIMA_2 = Arima(xts_list_Diff$TRON_ts, order=c(0,0,0))
Tron_model_ARIMA_3 = Arima(xts_list_Diff$TRON_ts, order=c(1,0,1))

AIC(Tron_model_ARIMA_1)
AIC(Tron_model_ARIMA_2)
AIC(Tron_model_ARIMA_3)

BIC(Tron_model_ARIMA_1)
BIC(Tron_model_ARIMA_2)
BIC(Tron_model_ARIMA_3)

## ARIMA (5,0,4) and (0,0,0) have the same AIC.
## ARIMA (0,0,0) is the one with the best BIC.
# Model 2 will be chosen as it has the best BIC and the same BIC.


#Litecoin
ARIMA_litecoin_ts #Graph indicates the same (1,0,1)

Litecoin_model_ARIMA_1 = Arima(xts_list_Diff$litecoin_ts, order=c(1,0,1))
Litecoin_model_ARIMA_2 = Arima(xts_list_Diff$litecoin_ts, order=c(1,0,0))
Litecoin_model_ARIMA_3 = Arima(xts_list_Diff$litecoin_ts, order=c(0,0,1))

AIC(Litecoin_model_ARIMA_1)
AIC(Litecoin_model_ARIMA_2)
AIC(Litecoin_model_ARIMA_3)

BIC(Litecoin_model_ARIMA_1)
BIC(Litecoin_model_ARIMA_2)
BIC(Litecoin_model_ARIMA_3)

## ARIMA (1,0,0) and (0,0,1) have the same AIC.
```

```
## ARIMA (1,0,0) and (0,0,1) have the same BIC.
# As the difference is quite small, model 1 will be chosen as it has AR &
MA parameters.

#Chainlink
ARIMA_chainlink_ts #Graph indicates (2,0,2) instead of (1,0,2).

Chainlink_model_ARIMA_1 = Arima(xts_list_Diff$chainlink_ts, order=c(1,0,2))
Chainlink_model_ARIMA_2 = Arima(xts_list_Diff$chainlink_ts, order=c(2,0,2))
Chainlink_model_ARIMA_3 = Arima(xts_list_Diff$chainlink_ts, order=c(1,0,1))

AIC(Chainlink_model_ARIMA_1)
AIC(Chainlink_model_ARIMA_2)
AIC(Chainlink_model_ARIMA_3)

BIC(Chainlink_model_ARIMA_1)
BIC(Chainlink_model_ARIMA_2)
BIC(Chainlink_model_ARIMA_3)

## ARIMA (1,0,2) has the best AIC.
## ARIMA (1,0,1) has the best BIC.
# model 3 will be chosen as it has lower parameters.

###### GARCH Model ######

## Testing for ARCH Effects ##
#Bitcoin

arch.test(bitcoin_model_ARIMA_3$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#Ethereum

arch.test(ethereum_model_ARIMA_3$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#Theter

arch.test(tether_model_ARIMA_1$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#BNB

arch.test(BNB_model_ARIMA_2$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#XRP

arch.test(XRP_model_ARIMA_2$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#Cardano

arch.test(Cardano_model_ARIMA_2$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#Dogecoin

arch.test(Dogecoin_model_ARIMA_1$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)
```

```
#TRON

arch.test(Tron_model_ARIMA_2$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#Litecoin

arch.test(Litecoin_model_ARIMA_1$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)

#ChainLink

arch.test(Chainlink_model_ARIMA_3$residuals,arch = c("box","Lm"),alpha =
0.05,lag.max = 5)


#### Getting the best GARCH Model for each Crypto ####

#Bitcoin

bitcoin_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="sGARCH"),distribution.model = "norm")
bitcoin_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
bitcoin_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
bitcoin_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="apARCH"),distribution.model = "norm")
bitcoin_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="eGARCH"),distribution.model = "norm")
bitcoin_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="csGARCH"),distribution.model = "norm")
bitcoin_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,0)),variance.model = list(model="iGARCH"),distribution.model = "norm")


bitcoin_model_garch_s = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_sGarch)
bitcoin_model_garch_t = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_tGarch)
bitcoin_model_garch_gjr = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_gjrGarch)
bitcoin_model_garch_ap = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_apArch)
bitcoin_model_garch_e = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_eGarch,solver = 'hybrid')
bitcoin_model_garch_cs = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_csGarch)
bitcoin_model_garch_i = ugarchfit(data = xts_list_Diff$bitcoin_ts,spec =
bitcoin_spec_iGarch)

#Ethereum
ethereum_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="sGARCH"),distribution.model = "norm")
ethereum_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
```

```
ethereum_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
ethereum_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="apARCH"),distribution.model = "norm")
ethereum_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="eGARCH"),distribution.model = "norm")
ethereum_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="csGARCH"),distribution.model = "norm")
ethereum_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="iGARCH"),distribution.model = "norm")


ethereum_model_garch_s = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_sGarch)
ethereum_model_garch_t = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_tGarch)
ethereum_model_garch_gjr = ugarchfit(data = xts_list_Diff$ethereum_ts,spec
= ethereum_spec_gjrGarch)
ethereum_model_garch_ap = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_apArch)
ethereum_model_garch_e = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_eGarch,solver = 'hybrid')
ethereum_model_garch_cs = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_csGarch)
ethereum_model_garch_i = ugarchfit(data = xts_list_Diff$ethereum_ts,spec =
ethereum_spec_iGarch)


#Tether
tether_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="sGARCH"),distribution.model = "norm")
tether_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
tether_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
tether_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="apARCH"),distribution.model = "norm")
tether_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="eGARCH"),distribution.model = "norm")
tether_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="csGARCH"),distribution.model = "norm")
tether_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(5,5)),variance.model = list(model="iGARCH"),distribution.model = "norm")


tether_model_garch_s = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_sGarch)
tether_model_garch_t = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_tGarch)
tether_model_garch_gjr = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_gjrGarch)
tether_model_garch_ap = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_apArch)
tether_model_garch_e = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_eGarch,solver = 'hybrid')
tether_model_garch_cs = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_csGarch)
```

```
tether_model_garch_i = ugarchfit(data = xts_list_Diff$tether_ts,spec =
tether_spec_iGarch)

#BNB
BNB_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="sGARCH"),distribution.model = "norm")
BNB_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
BNB_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
BNB_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="apARCH"),distribution.model = "norm")
BNB_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="eGARCH"),distribution.model = "norm")
BNB_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="csGARCH"),distribution.model = "norm")
BNB_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="iGARCH"),distribution.model = "norm")


BNB_model_garch_s = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_sGarch)
BNB_model_garch_t = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_tGarch)
BNB_model_garch_gjr = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_gjrGarch)
BNB_model_garch_ap = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_apArch)
BNB_model_garch_e = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_eGarch,solver = 'hybrid')
BNB_model_garch_cs = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_csGarch)
BNB_model_garch_i = ugarchfit(data = xts_list_Diff$BNB_ts,spec =
BNB_spec_iGarch)

#XRP
XRP_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="sGARCH"),distribution.model = "norm")
XRP_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
XRP_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
XRP_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="apARCH"),distribution.model = "norm")
XRP_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="eGARCH"),distribution.model = "norm")
XRP_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="csGARCH"),distribution.model = "norm")
XRP_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="iGARCH"),distribution.model = "norm")


XRP_model_garch_s = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_sGarch)
XRP_model_garch_t = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_tGarch)
```

```
XRP_model_garch_gjr = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_gjrGarch)
XRP_model_garch_ap = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_apArch)
XRP_model_garch_e = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_eGarch,solver = 'hybrid')
XRP_model_garch_cs = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_csGarch)
XRP_model_garch_i = ugarchfit(data = xts_list_Diff$XRP_ts,spec =
XRP_spec_iGarch)

#Cardano
Cardano_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="sGARCH"),distribution.model = "norm")
Cardano_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
Cardano_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
Cardano_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="apARCH"),distribution.model = "norm")
Cardano_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="eGARCH"),distribution.model = "norm")
Cardano_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="csGARCH"),distribution.model = "norm")
Cardano_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="iGARCH"),distribution.model = "norm")


Cardano_model_garch_s = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_sGarch)
Cardano_model_garch_t = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_tGarch)
Cardano_model_garch_gjr = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_gjrGarch)
Cardano_model_garch_ap = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_apArch)
Cardano_model_garch_e = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_eGarch,solver = 'hybrid')
Cardano_model_garch_cs = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_csGarch)
Cardano_model_garch_i = ugarchfit(data = xts_list_Diff$cardano_ts,spec =
Cardano_spec_iGarch)


#Dogecoin
Dogecoin_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="sGARCH"),distribution.model = "norm")
Dogecoin_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
Dogecoin_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
Dogecoin_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="apARCH"),distribution.model = "norm")
Dogecoin_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="eGARCH"),distribution.model = "norm")
Dogecoin_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="csGARCH"),distribution.model = "norm")
```

```
Dogecoin_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="iGARCH"),distribution.model = "norm")


Dogecoin_model_garch_s = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_sGarch)
Dogecoin_model_garch_t = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_tGarch)
Dogecoin_model_garch_gjr = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec
= Dogecoin_spec_gjrGarch)
Dogecoin_model_garch_ap = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_apArch)
Dogecoin_model_garch_e = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_eGarch,solver = 'hybrid')
Dogecoin_model_garch_cs = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_csGarch)
Dogecoin_model_garch_i = ugarchfit(data = xts_list_Diff$dogecoin_ts,spec =
Dogecoin_spec_iGarch)


#TRON
TRON_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="sGARCH"),distribution.model = "norm")
TRON_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
TRON_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
TRON_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="apARCH"),distribution.model = "norm")
TRON_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="eGARCH"),distribution.model = "norm")
TRON_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="csGARCH"),distribution.model = "norm")
TRON_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(0,0)),variance.model = list(model="iGARCH"),distribution.model = "norm")


TRON_model_garch_s = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_sGarch)
TRON_model_garch_t = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_tGarch)
TRON_model_garch_gjr = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_gjrGarch)
TRON_model_garch_ap = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_apArch)
TRON_model_garch_e = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_eGarch,solver = 'hybrid')
TRON_model_garch_cs = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_csGarch)
TRON_model_garch_i = ugarchfit(data = xts_list_Diff$TRON_ts,spec =
TRON_spec_iGarch)


#Litecoin
Litecoin_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="sGARCH"),distribution.model = "norm")
Litecoin_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
```

```
Litecoin_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
Litecoin_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="apARCH"),distribution.model = "norm")
Litecoin_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="eGARCH"),distribution.model = "norm")
Litecoin_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="csGARCH"),distribution.model = "norm")
Litecoin_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="iGARCH"),distribution.model = "norm")


Litecoin_model_garch_s = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_sGarch)
Litecoin_model_garch_t = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_tGarch)
Litecoin_model_garch_gjr = ugarchfit(data = xts_list_Diff$litecoin_ts,spec
= Litecoin_spec_gjrGarch)
Litecoin_model_garch_ap = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_apArch,solver = 'hybrid')
Litecoin_model_garch_e = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_eGarch,solver = 'hybrid')
Litecoin_model_garch_cs = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_csGarch)
Litecoin_model_garch_i = ugarchfit(data = xts_list_Diff$litecoin_ts,spec =
Litecoin_spec_iGarch)


#ChainLink
Chainlink_spec_sGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="sGARCH"),distribution.model = "norm")
Chainlink_spec_tGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="fGARCH",garchOrder = c(1, 1),submodel
= "TGARCH"),distribution.model = "norm")
Chainlink_spec_gjrGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="gjrGARCH"),distribution.model =
"norm")
Chainlink_spec_apArch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="apARCH"),distribution.model = "norm")
Chainlink_spec_eGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="eGARCH"),distribution.model = "norm")
Chainlink_spec_csGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="csGARCH"),distribution.model = "norm")
Chainlink_spec_iGarch = ugarchspec(mean.model = list(armaOrder =
c(1,1)),variance.model = list(model="iGARCH"),distribution.model = "norm")


Chainlink_model_garch_s = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_sGarch)
Chainlink_model_garch_t = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_tGarch)
Chainlink_model_garch_gjr = ugarchfit(data =
xts_list_Diff$chainlink_ts,spec = Chainlink_spec_gjrGarch)
Chainlink_model_garch_ap = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_apArch)
Chainlink_model_garch_e = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_eGarch,solver = 'hybrid')
Chainlink_model_garch_cs = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_csGarch)
```

```
Chainlink_model_garch_i = ugarchfit(data = xts_list_Diff$chainlink_ts,spec
= Chainlink_spec_iGarch)

####### Test out-of-sample ###########

## Getting values ##
crypto_2023 = crypto_history(coin_list = coins, convert = "USD", limit =
NULL, start_date = "20230101", end_date = "20231231", interval = NULL,
sleep = 60, finalWait = TRUE)

## Selecting the determined columns
crypto_2023 = select(crypto_2023,c("timestamp","name","symbol","close"))

## Adjusting Values
crypto_2023$close = as.numeric(str_replace(crypto_2023$close, ',', '.'))
crypto_2023$timestamp = as.Date(crypto_2023$timestamp)
crypto_2023 <- crypto_2023 %>%
  mutate(name = ifelse(name == "Tether USDt", "Tether", name))

coins <- coins %>%
  mutate(name = ifelse(name == "Tether USDt", "Tether", name))

## Getting a DataFrame for each Crypto ##
i = 1

for (i in 1:10) {

  assign(paste(coins$name[i],"_df_2023",sep = ""), filter(crypto_2023,
crypto_2023$name == coins$name[i]))


}

## Creating time series for each Crypto and Getting the statistics ##

bitcoin_ts_2023 <- xts(Bitcoin_df_2023$close,
order.by=Bitcoin_df_2023$timestamp)
ethereum_ts_2023 <- xts(Ethereum_df_2023$close, order.by =
Ethereum_df_2023$timestamp)
tether_ts_2023 <- xts(Tether_df_2023$close, order.by =
Tether_df_2023$timestamp)
BNB_ts_2023 <- xts(BNB_df_2023$close, order.by = BNB_df_2023$timestamp)
XRP_ts_2023 <- xts(XRP_df_2023$close, order.by = XRP_df_2023$timestamp)
cardano_ts_2023 <- xts(Cardano_df_2023$close, order.by =
Cardano_df_2023$timestamp)
dogecoin_ts_2023 <- xts(Dogecoin_df_2023$close, order.by =
Dogecoin_df_2023$timestamp)
TRON_ts_2023 <- xts(TRON_df_2023$close, order.by = TRON_df_2023$timestamp)
litecoin_ts_2023 <- xts(Litecoin_df_2023$close, order.by =
Litecoin_df_2023$timestamp)
chainlink_ts_2023 <- xts(Ethereum_df_2023$close, order.by =
Ethereum_df_2023$timestamp)


### Plotting and getting the statistics + Stationarity test (ADF) ##

xts_list_2023 =
merge(bitcoin_ts_2023,ethereum_ts_2023,tether_ts_2023,BNB_ts_2023,XRP_ts_20
23,cardano_ts_2023,dogecoin_ts_2023,TRON_ts_2023,litecoin_ts_2023,chainlink
_ts_2023)
```

```
i = 1
j = ncol(xts_list_2023)
for (i in 1:j) {

  print(plot.xts(xts_list_2023[,i],main = colnames(xts_list_2023[,i])))
  assign(paste("summary_2023",colnames(xts_list_2023[,i]),sep =
""),summary(xts_list_2023[,i]))
  assign(paste("adf_2023",colnames(xts_list_2023[,i]),sep =
""),adf.test(xts_list_2023[,i], alternative = "stationary", k=0)) # p-value
< 0.05 indicates the TS is stationary

}

### Listing the coins to make it easier to run the codes + Differentiating
them, to make it stationary ###
#xts_list_Diff_2023 = diff(xts_list_2023, lag = 1)
xts_list_Diff_2023 = diff(log(xts_list_2023), lag = 1)
xts_list_Diff_2023 = xts_list_Diff_2023[-1,]



### Forecast ###

crypto_names <- tolower(unique(crypto$name))
spec_names <- c("sGarch", "tGarch", "gjrGarch", "apArch", "eGarch",
"csGarch", "iGarch")

names(xts_list_Diff_2023) = tolower(names(xts_list_Diff_2023))

results_list <- list()

for (crypto_name in crypto_names) {

  for (spec_name in spec_names) {

    # Constructing variable names based on crypto_name
    crypto_column <- paste0(crypto_name, "_ts_2023")
    crypto_data <- xts_list_Diff_2023[, crypto_column]
    spec <- get(paste0(crypto_name,"_spec_", spec_name))
    fit_name <- paste0(crypto_name, "_model_out_sample", spec_name)

    # Fit the model
    assign(fit_name, ugarchfit(data = crypto_data, spec = spec,solver =
'hybrid'))

    # Forecast
    ugfore_name <- paste0(crypto_name, "_forecast_", spec_name)
    assign(ugfore_name, ugarchforecast(fitORspec =
get(paste0(crypto_name,"_model_", spec_name)), n.ahead = 30))

    # Extract actual volatility
    actual_volatility_name <- paste0(crypto_name, "_actual_volatility_",
spec_name)
    assign(actual_volatility_name, sigma(get(fit_name)))

    # Subset actual volatility
    subset_vol_name <- paste0(crypto_name, "_subset_vol_", spec_name)
    assign(subset_vol_name, get(actual_volatility_name)[1:30])

    # Calculate RMSE
    mse_name <- paste0(crypto_name, "_mse_", spec_name)
```

```
    rmse_name <- paste0(crypto_name, "_rmse_", spec_name)
    assign(mse_name, mean((sigma(get(paste0(ugfore_name))) -
get(subset_vol_name))^2))
    assign(rmse_name, sqrt(get(mse_name)))

    # Store results in a list
    result <- list(
      crypto_name = crypto_name,
      spec_number = spec_name,
      mse = get(mse_name),
      rmse = get(rmse_name)
    )

    results_list[[paste0(crypto_name, "_result_", spec_name)]] <- result

  }
}


# Combine results into a data frame
results_df <- do.call(rbind, results_list)
results_df = data.frame(results_df)
results_df$rmse <- lapply(results_df$rmse, function(x) as.numeric(x))

best_specs <- results_df %>%
  group_by(crypto_name) %>%
  slice(which.min(rmse)) %>%
  ungroup()

best_specs_rank <- results_df %>%
  group_by(crypto_name) %>%
  mutate(rank = rank(unlist(rmse))) %>%
  ungroup()
```