CRYPTO-STATISTICS: AN EXPLORATION OF THE BEHAVIOR OF CRYPTO MARKET AND OTHER RELATED PROBLEMS

by

MINJIE YU

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN APPLIED MATHEMATICAL SCIENCES

2026

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Prof. Ravindra Khattree, Ph.D., Chair
Prof. Seong-Yeon Cho, Ph.D.
Prof. Subbaiah Perla, Ph.D.
Prof. Harvey Qu, Ph.D.
Prof. Nalini Ravishanker, Ph.D.

To my family,

thank you for your endless support and belief in me.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Prof. Ravindra Khattree, for his continuous guidance, mentorship, encouragement, and expertise throughout every stage of this dissertation. His steadfast support of my research ideas, his insightful direction on statistical methodology, and his thoughtful suggestions on the study's development have been invaluable. I am especially grateful for the meticulous care and countless hours he devoted to supporting every detail of this work.

I would also like to thank my committee members, Prof. Seong-Yeon Cho, Prof. Subbaiah Perla, Prof. Harvey Qu and Prof. Nalini Ravishanker, for their thoughtful feedback and constructive insights, which have greatly strengthened this research. I am grateful to my manager, Yong Li, for facilitating access to the data used in this research and for his encouragement, understanding and support throughout my doctoral journey.

Finally, I extend my heartfelt thanks to my family for their love, understanding, encouragement and unwavering belief in me during this journey. A special thanks goes to my son, Elliott, whose companionship, cheerful encouragement and the countless quiet moments we shared studying together brought me strength and motivation.

Minjie Yu

ABSTRACT

CRYPTO-STATISTICS: AN EXPLORATION OF THE BEHAVIOR OF CRYPTO MARKET AND OTHER RELATED PROBLEMS

by

Minjie Yu

Adviser: Prof. Ravindra Khattree, Ph.D.

This dissertation empirically explores the statistical behavior and predictive dynamics of major cryptocurrencies including Bitcoin, Ethereum, Ethereum Classic, XRP, Cardano, Dogecoin, and Tether. Using the daily data on crypto prices spanning from 2017 to 2024, the study integrates various classical and modern statistical methodologies to analyze the return distributions, interdependencies, predictive modeling and strategic methods in the crypto market. Descriptive and multivariate analyses, including Principal Component Analysis, Factor Analysis and Dynamic Factor Analysis, are conducted to uncover various empirical properties and reduce dimensionality.

The study develops predictive models for both present and future returns, daily and weekly returns of cryptocurrencies, using a diverse set of financial indices, including forex rates, stock indices, and commodity prices, as predictors. A wide range of regression techniques is applied, including full model estimation, submodel selection via AIC, BIC, and LASSO, and advanced shrinkage methods such as Ridge, Elastic Net, Adaptive LASSO, and SCAD. Post-shrinkage estimators are introduced to enhance model robustness and predictive accuracy. The models are evaluated using the bootstrap-based prediction error metrics.

A novel contribution of this study is the development and evaluation of pair-trading strategies based on cointegration analysis and spread thresholds. These strategies are tested across various cryptocurrency pairs, including Bitcoin-Ethereum, Ethereum-Ethereum Classic and Bitcoin-Ethereum Classic, with usual regression as well as the Orthogonal Distance Regression. The profitability of several different entry and exit rules is evaluated using historical and by using the future out-of-sample period datasets.

To further understand risk dynamics, the study applies univariate and multivariate GARCH models to investigates the time-varying volatility dynamics of major cryptocurrencies. Univariate GARCH models are fitted to each cryptocurrency to characterize individual volatility processes, revealing strong ARCH and GARCH effects as well as persistent volatility patterns. The multivariate BEKK model is estimated for Bitcoin, Ethereum, and Ethereum Classic, demonstrating significant volatility spillovers and dynamic correlations among these assets. These findings highlight the interconnected, highly reactive, and evolving nature of volatility within cryptocurrency markets.

This dissertation also extends its modeling framework to a real-world problem of predicting daily power supply interruptions using weather, reliability program, and seasonal data collected at DTE Energy. The findings demonstrate the effectiveness of post-shrinkage strategies and transformation techniques in improving predictive performance.

Keywords: Cryptocurrency, Energy supply, Pair trading, Post-shrinkage estimator

AMS Subject Classification: 62J07, 91B84, 62M10, 62H12, 62H25, 91G15, 62P05, 62P30

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

LIST OF TABLES

LIST OF TABLES—Continued

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION AND STATISTICAL PRELIMINARIES

The cryptocurrency market has emerged as a significant component of global finance, distinguished by its decentralization, high volatility, and rapid innovation. As crypto assets continue to gain traction among investors, institutions, and policymakers, there is a growing interest in understanding their statistical behavior, interdependencies, and the potential for predictive modeling and strategic trading. This dissertation aims to investigate the statistical properties and predictive dynamics of certain major cryptocurrencies. The study spans multiple analytical dimensions, including descriptive statistics, multivariate analysis, regression modeling, short tern prediction and cointegration-based trading strategies. Additionally, as a somewhat unrelated energy problem the post-shrinkage methodologies applied in the predictive modelings are extended to a real-world utility application involving the prediction of daily power supply interruptions, demonstrating the broader applicability of the proposed statistical framework beyond the domain of digital finance.

The dissertation is organized into eight chapters, each addressing a distinct analytical problem. Chapter two presents a comprehensive descriptive analysis of seven major cryptocurrencies: Bitcoin, Ethereum, Ethereum Classic, XRP, Cardano, Dogecoin, and Tether, using daily data from December 31, 2017, to June 17, 2024. The chapter evaluates the return and price distributions, identifies outliers, explores interrelationships among assets, and assesses normality assumptions through formal statistical tests and visual diagnostics. Various measures of association, such as Pearson, Spearman, Kendall, and Chatterjee correlations, are employed to assess linear and nonlinear dependencies

among various assets. These analyses establish a foundational understanding of the distributional and relational characteristics of crypto returns.

Chapter three extends the descriptive analysis to a multivariate framework, employing Principal Component Analysis (PCA), Factor Analysis (FA) and Dynamic Factor Analysis to uncover latent structures and reduce dimensionalities in both standardized price and daily return data. These techniques reveal clusters of cryptocurrencies with similar behavioral patterns, identify dominant components that explain the majority of variance in the data and show how the shared drivers evolove over time, providing a robust framework for understanding the structural relationships and latent dimensions within the cryptocurrency market.

Chapter four focuses on the prediction of daily returns for each of the cryptocurrency. Using a diverse set of financial indicators including foreign exchange rates, various stock and commodity indices alongside other crypto returns, the study develops regression models for both present-day and one-day-ahead predictions. A wide range of estimation techniques is applied, including full model estimation, submodel selection via AIC, BIC, and LASSO, and penalized methods such as Ridge, Elastic Net, Adaptive LASSO, and SCAD. Post-shrinkage estimators are introduced to enhance model robustness and predictive accuracy. The PCA is utilized to reduce dimensionality and mitigate multicollinearity in the regressions. The model performances are evaluated using bootstrap-based prediction error metrics to ensure reliable comparisons.

Chapter five extends the predictive modeling framework to weekly returns, offering a broader temporal perspective. The analysis incorporates both current and lagged weekly returns of cryptocurrencies and financial indices to capture short-term dependencies and autocorrelations. The modeling strategy builds upon the framework

established in Chapter four, with adaptations to accommodate longer forecasting horizons and assess the stability of predictive relationships.

Chapter six explores pair-trading strategies for certain cyrpto pairs based on cointegration analysis. The chapter identifies cryptocurrency pairs with long-term equilibrium relationships and develops trading strategies based on spread thresholds. Orthogonal distance regression is also employed to refine trading signals, and the profitability of different entry and exit rules is evaluated using historical and out-of-sample data. This chapter demonstrates the practical utility of statistical modeling in informing trading decisions within crypto markets.

Chapter seven examines the time-varying volatility dynamics of major cryptocurrencies using both univariate and multivariate GARCH models. The chapter highlights several main features of cryptocurrency returns, including pronounced volatility clustering, strong persistence of shocks, and heavy-tailed return distributions. GARCH-type models offer a coherent and effective framework for capturing the volatility characteristics of cryptocurrencies.

Chapter eight demonstrates the application of the modeling framework to a real-world utility problem of predicting daily power supply interruptions. Using data on weather conditions, reliability program, and seasonal factors, the chapter applies the predictive strategies including full model estimation, submodel selection, penalized regression methods and post-shrinkage estimators to identify the optimal set of predictors and estimate corresponding coefficients, and leverage the Copula transformation to induce normality in the data, for accurately predicting the number of interruptions and provide insights into the effectiveness of programs in improving the service to the energy customers. Although not directly related to previous chapters, this problem is not only

important in its own right but also relevant to cryptos, mining of which requires a great volume of power supply.

The financial datasets used in this study are primary sourced from publicly available platforms. Cryptocurrency data were obtained from Yahoo Finance and include daily open, close, high, and low prices for Bitcoin, Ethereum, Ethereum Classic, XRP, Cardano, Dogecoin, and Tether. In addition to crypto data, the study incorporates daily closing prices for 20 financial indices querying from Yahoo Finance and generating via TC2000 software. These indices include 12 major forex pairs, 5 stock market indices, and 3 commodity indices, all serving as explanatory variables in predictive models to assess macroeconomic influences on crypto returns. The power supply interruption data were sourced from the DTE Electric company internal database.

The remainder of this chapter introduces the statistical methodologies employed in the modeling and analysis throughout the dissertation. The techniques discussed include full model estimation, submodel selection, penalized regression methods, post-shrinkage estimators, copula-based normalization, the Chatterjee correlation coefficient, and cointegration analysis and its application to pair-trading strategies.

1.1 Post-Shrinkage Strategy and Estimation Method

We will first describe the basic model and its variations. With a multitude of variables, one has to be mindful of the strength of signals provided by each of these variables. For improved estimation, these in turn will determine the submodels. The idea is to not ignore moderate or weak signals but to use them to improve the model. This is where our shrinkage strategies lie. The number of variables is as such quite large and for simplicity, we will not include any polynomial or cross-product (interaction) terms. Many variables are categorical and they will be treated as such. There may be multicollinearity among variables but since the objective here is not to select or discard variables or to

define new transformations, to save the space, we will not address these aspects of the problem here. Our analyses will address various penalized and post- shrinkage strategies only. A few good references for the above mentioned techniques are Ahmed et al.(2023) , Zou and Hastie (2005),Ahmed (2014) , Zou (2006) , Ahmed (2021), Arashi et al.(2021), Gao et al.(2017) .

1.1.1 The Full Model Estimator

To start with, the basic model relating a vector of response $\mathbf{Y}$ to all predictors is the standard linear model

$$\mathbf{Y}_{n\times 1} = \mathbf{X}_{n\times p}\beta_{p\times 1} + \varepsilon_{n\times 1}, \;\; Rank(\mathbf{X}) = p < n. \tag{1.1}$$

With number of predictors defined, the matrix $\mathbf{X}$ is a $n \times p$ matrix of $p-1$ predictors whose values are arranged in all but 0^{th} column. The 0^{th} column is a constant vector $\mathbf{1}_n = (1,1,\ldots,1)'$. The vector $\mathbf{Y}$ is the vector of dependent variable, β is the unknown parameters representing intercept and slopes, ε is the $n \times 1$ vector of random errors. We assume errors to be identically and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon^T) = \sigma^2 I_n$, where $\sigma^2 > 0$. The matrix $\mathbf{X}$ is assumed to be of full column rank $p < n$.

The Equation (1.1) is what we call as the *full model*. The method of maximum likelihood is used to estimate the coefficients for various predictor variables. Some of the predictors may not provide strong signals for the response variable and thus it may be necessary to modify the model either by eliminating some of the predictors (thereby obtaining a submodel) on by deemphasizing the roles of these weak predictors by using various specialized techniques. Let $\mathbf{X} = \begin{bmatrix}\mathbf{X}_1 : & \mathbf{X}_2\end{bmatrix}$ and $\beta = \begin{bmatrix}\beta_1 \\ \beta_2\end{bmatrix}$.Then the Equation (1.1) can be written as

$$\mathbf{Y} = \mathbf{X}_1\beta_1 + \mathbf{X}_2\beta_2 + \varepsilon \tag{1.2}$$

where β_1, along with an intercept, is the vector of parameters for variables corresponding to matrix $\mathbf{X}_1$ that may be retained in the submodel. If we assume that $\beta_2 = \mathbf{0}$ then the reduced model is

$$\mathbf{Y} = \mathbf{X}_1\beta_1 + \varepsilon \ . \tag{1.3}$$

By assuming $\beta_2 = \mathbf{0}$ the model (1.3) corresponds to the assumption that predictors corresponding to matrix $\mathbf{X}_2$ are unimportant. This decision to drop these variables may be arrived at through various variable selection approaches.

Under model in Equation (1.1) and under the assumption of normality, stated earlier, the maximum likelihood estimator of β is

$$\hat{\beta}^{FM} = \begin{bmatrix} \hat{\beta}_1^{\,FM} \\ \hat{\beta}_2^{\,FM} \end{bmatrix} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}. \tag{1.4}$$

It is easy to verify that the full model estimator of β_1 can be written as,

$$\hat{\beta}_1^{\,FM} = (\mathbf{X}_1'S\mathbf{X}_1)^{-1}\mathbf{X}_1'S\mathbf{Y} \tag{1.5}$$

where $\mathbf{S} = I - \mathbf{X}_2(\mathbf{X}_2'\mathbf{X}_2)^{-1}\mathbf{X}_2'$. Also under model in Equation (1.3), the maximum likelihood estimator of β_1 is given by,

$$\hat{\beta}_1^{\,SM} = (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'\mathbf{Y}. \tag{1.6}$$

We will employ a variety of estimation techniques depending on how we regularize the parameters. Various approaches of regularization and corresponding criteria are discussed next.

1.1.2 Submodel Selection

Numerous model selection techniques have been developed to effectively eliminate predictors with sparse signals and retain only the most significant variables, resulting in a refined submodel.

The classical variable selection methods rely on the full model estimators and employ step-wise regression with AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) criteria to identify a handful of highly influential predictors. In this analysis, we will employ the criteria of AIC, BIC and LASSO to select submodels. The corresponding statistics are defined below.

1. $AIC = 2k - 2\ln(\hat{L})$,

   where k is the number of estimated parameters in the model, $\hat{L}$ is the maximized value of the likelihood function for the model. In the AIC approach for model selection, we compare different candidate models based on their AIC values. We prefer the models with the smaller AIC values, as it suggests a better trade-off between goodness of fit and model complexity (number of parameters). We will thus rely on smallest value of AIC.

2. $BIC = k\ln(n) - 2\ln(\hat{L})$,

   where n is the number of observations, k and $\hat{L}$ are same as defined above. The BIC criterion is similar to the AIC in that we compute the BIC value for each candidate model, and select the model with lowest BIC value as the best model. Compared to

AIC, the BIC introduces a stronger penalty for models with more parameters, as instead of $2k$ it uses $k\ln(n)$ in the first term. This helps prevent any over-fitting in some cases.

3. $\hat{\beta}^{LASSO} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$.

   In the LASSO solution, sufficient small coefficients are shrunk towards zero and hence are dropped from the model. The resulting submodel is the one selected by LASSO. In essence, the second term in the above expression acts as penalty for the large values of coefficients β_j.

1.1.3 Penalized Methods

Additionally, various penalized model selection methods (similar to but not same as the LASSO method included above), may be employed to obtained stable prediction equations. We will describe below some of those which are used in this work, see Ahmed et al.(2023) .

3. The LASSO (least absolute shrinkage and selection operator) is a widely utilized penalized method that leverages an L_1-norm penalty to regularize regression parameters. Mathematically speaking,

$$\hat{\beta}^{LASSO} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{1.7}$$

   The first term in above is the sum of squared errors and the the second term is a L_1-norm penalty function. The parameter λ is suitably selected through a grid search, where the value that minimizes the BIC is chosen for this analysis. In the LASSO estimation, sufficiently small coefficients are shrunk towards zero. The

same approach is used for choosing the tuning parameter λ (or λ_1, λ_2) for Ridge, ENET, and ALASSO described below.

4. The Ridge estimation shrinks the coefficients by using a L_2-norm penalty and minimizes the sum of squared error. Specifically,

$$\hat{\beta}^{Ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{1.8}$$

5. The ENET (elastic net) regularization is introduced as another method which is especially effective when some of the independent variables are highly correlated, see Zou and Hastie (2005). It combines a L_1 penalty term and L_2 penalty term in the objection function and thus,

$$\hat{\beta}^{ENET} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{1.9}$$

6. The ALASSO (Adaptive LASSO) method modifies the LASSO estimator to use adaptive weights on the L_1 penalty. It is,

$$\hat{\beta}^{ALASSO} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \hat{\omega}_j |\beta_j| \right\}. \tag{1.10}$$

   It is known that this estimator has the property of model consistency. See Zou (2006) .

7. The SCAD (Smoothly Clipped Absolute Deviation) method combines the favorable characteristics of subset selection and ridge regression, and produces sparse solutions that effectively addresses variable selection, see details in Fan and Li (2001) . The estimator is derived as

$$\hat{\beta}^{SCAD} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \mathbf{X}_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} p_{\alpha,\lambda}(|\beta_j|) \right\}. \tag{1.11}$$

where $p_{\alpha,\lambda}(.)$ is the smoothly clipped absolute deviation penalty function. This penalty function leaves large values of θ not excessively penalized and makes the solution continuous, defined by,

$$p'_{\alpha,\lambda}(\theta) = \lambda \left\{ I(\theta \leq \lambda) + \frac{(\alpha\lambda - \theta)_+}{(\alpha - 1)\lambda} I(\theta > \lambda) \right\} \tag{1.12}$$

where $\lambda > 0$ and $\alpha > 2$ are the tuning parameters which are optimized through the grid search aiming to minimize BIC. In this analysis, $\theta > 0$, and the function $f(z) = z_+$ is equal to z if $z \geq 0$ and 0 otherwise.

1.1.4 Post-Shrinkage Estimators

The post-shrinkage estimators are useful when a model has some strong signals, a few moderate signals and many weak signals. It combines estimates from the full model with the estimates from a submodel to improve the estimation of the submodel.

The estimator in Equation (1.6) ignores all information available in submatrix $\mathbf{X}_2$. The corresponding variables may or may not be important depending on whether or not the model is overfitted (i.e whether $\beta_2 \neq 0$ or $\beta_2 = 0$). To this end, we may want to consider the idea from James and Stein (1992) and define a shrinkage estimator of β_1 by utilizing a smooth function of a corresponding test statistics for testing $\beta_2 \neq 0$ and by shrinking the full model estimator towards the direction of submodel estimator as,

$$\hat{\beta_1}^{S} = \hat{\beta_1}^{SM} + \left(\hat{\beta_1}^{FM} - \hat{\beta_1}^{SM} \right) \left(1 - (p_2 - 2)\tau_n^{-1} \right), p_2 > 3 \tag{1.13}$$

where the test statistics τ_n is defined as $\tau_n = \frac{n}{\hat{\sigma}^2}(\hat{\beta_2}^{LSE})\mathbf{X}_2'S_1\mathbf{X}_2(\hat{\beta_2}^{LSE})$, $\hat{\beta_2}^{LSE} = (\mathbf{X}_2'S_1\mathbf{X}_2)^{-1}\mathbf{X}_2'S_1\mathbf{Y}$ and $S_1 = I - \mathbf{X}_1(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'$. Also $\hat{\sigma}^2$ is a consistent estimator of σ^2. See Ahmed (2014) .

Another estimator similar to that in Equation (1.13) is the positive shrinkage estimator. It was introduced to control the over-shrinking of a shrinkage estimator which may result in reversing the sign of certain estimates in the full model in some cases. It is defined as,

$$\hat{\beta_1}^{PS} = \hat{\beta_1}^{SM} + \left(\hat{\beta_1}^{FM} - \hat{\beta_1}^{SM}\right)\left(1 - (p_2 - 2)\tau_n^{-1}\right)^+ \tag{1.14}$$

where $z^+ = max(0, z)$.

Our approach will be to apply all of the above estimation methods to our data and examine which of these result in superior prediction of response variable Y. Later, we also adopt an approach based on a copula-transformation of the data on quantitative variables. See section 3 of this chapter.

1.2 Prediction comparison

A comparison of estimators given by Equations (1.7) - (1.14) is warranted. The mean prediction error (PE) of a model using an estimator $\hat{\beta_1^*}$ is defined as,

$$PE(\hat{\beta_1^*}) = \frac{1}{n}\left(\mathbf{Y} - \mathbf{X}_1\hat{\beta_1^*}\right)'\left(\mathbf{Y} - \mathbf{X}_1\hat{\beta_1^*}\right) \tag{1.15}$$

It is appropriate that to avoid any possible bias such evaluations are made independent of the data used to estimate the vector β. Thus the whole dataset is randomly divided into training data which are used to arrive at the corresponding estimator and the

test data which are used to evaluate the performance. Let them be $[\mathbf{Y}_{Training} : \mathbf{X}_{Training}]$ and $[\mathbf{Y}_{Test} : \mathbf{X}_{Test}]$. Accordingly, Equation (1.15) for the *test data* becomes,

$$PE(\hat{\beta}_1^*) = \frac{1}{n_{Test}} \left(\mathbf{Y}_{Test} - (\mathbf{X}_1)_{Test}\hat{\beta}_1^*\right)' \left(\mathbf{Y}_{Test} - (\mathbf{X}_1)_{Test}\hat{\beta}_1^*\right), \tag{1.16}$$

where $\mathbf{Y}_{Test}$ is the vector of response values in the test dataset, and $\hat{\beta}_1^*$ is a typical estimator obtained via one of the Equations (1.7) to (1.14) with $\mathbf{X}$ matrix replaced by $\mathbf{X}_{Training}$. To ensure a robust measure, we employ bootstrap sampling to calculate the average of the mean prediction errors for each estimator given by Equation (1.16) across all bootstrap samples. Mathematically, this can be expressed as,

$$AveragePE(\hat{\beta}_1^*) = \frac{1}{B}\sum_{i=1}^{B} PE_i(\hat{\beta}_1^*) \tag{1.17}$$

where B is the number of bootstrap iterations, and $PE_i(\hat{\beta}_1^*)$ represents the mean prediction error for the i-th bootstrap sample.

1.3 Copula Transformation

Bahuguna and Khattree (2020) have presented an approach to induce normality in the data by copula transformation under the assumption that data on continuous variables exhibit a dependence which can be approximated by a Gaussian copula. Thus, data from any multivariate distribution with multivariate cdf $F(.)$ and a Gaussian copula $C(.)$ can be transformed to multivariate normal distribution with cdf $G(.)$ via $C(.)$ as follows.

In essence, we are making an assumption that our data correspond to a Gaussian copula $G(\cdot) = \Phi_{(\boldsymbol{\mu},\, \boldsymbol{\Sigma})}(\cdot)$, In principle, the choices of mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ are arbitrary. Since our interest is in doing the multivariate analysis of dependence, we must choose $\boldsymbol{\Sigma}$ cautiously to retain the essential dependence features of

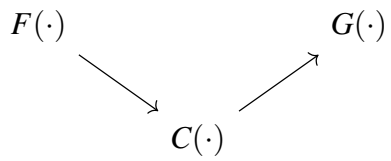


Figure 1.1: Transformation Through common copula (Here $F(\cdot)$ and $G(\cdot)$ are Cumulative distribution Functions and $C(\cdot)$ is Common Copula)

data. Since choice of $\boldsymbol{\mu}$ is often unimportant in such a situation, we will take its value to be zero vector. Likewise the variances in $\boldsymbol{\Sigma}$ will be taken as 1. As a graphical representation for a given distribution of data, say $\mathscr{D}$, our transformation thus works as

$$\mathscr{D} \longrightarrow \mathscr{U} \longrightarrow \mathscr{N}$$

and in a reverse direction as

$$\mathscr{N} \longrightarrow \mathscr{U} \longrightarrow \mathscr{D},$$

where $\mathscr{U}$ represents a multivariate distribution of cdfs (i.e. it is Copula) in which each marginal is $Uniform[0,1]$ and $\mathscr{N}$ represents a chosen multivariate normal distribution. Often, for convenience prescribed by zero means and variances 1, correlation may however be nonzero.

1.4 Chatterjee Correlation Coefficient

The Chatterjee correlation Coefficient, introduced by Chatterjee (2021) , provides a novel measure of statistical (presumably nonlinear) dependence between two random variables. In contrast to classical association measures such as Pearson, Spearman and Kendall correlations which primarily measure linear or monotonic relationships, the Chatterjee correlation is designed to detect general forms of dependence, potentially

nonlinear and non-monotonic structures. It is particularly effective for identifying complex patterns such as oscillatory signals. Moreover, it is known for its robustness to outliers and to the varying of the probability distributions. With its sophisticated capabilities, the Chatterjee correlation still retains a relatively simple formulation compared to certain similar dependence measures defined previously; See Sarkar and Ghosh (2018). We will formally define it for a sample first.

Let (X,Y) denote a pair of random variables, where Y is not a constant. Consider an i.i.d. pairs (X_i,Y_i), $i = 1,2,\ldots,n$, drawn from the joint distribution of (X,Y), with $n \geq 2$. Rearrange the data pairs in increasing order of X, so that

$$(X_{(1)},Y_{(1)}),(X_{(2)},Y_{(2)}),\ldots,(X_{(n)},Y_{(n)}), \quad \text{where } X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}.$$

If the X_i's and the Y_i's contain no ties, there is a unique way of doing the ordering, and the Chatterjee correlation ξ_n admits a relatively simpler formula,

$$\xi_n(X,Y) = 1 - \frac{3\sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1} \tag{1.18}$$

where r_i denotes the rank of $Y_{(i)}$ among $\{Y_{(1)},\ldots,Y_{(n)}\}$, and n is the sample size.

In the presence of ties, the general mathematical definition of the Chatterjee correlation coefficient is,

$$\xi_n(X,Y) = 1 - \frac{n\sum_{i=1}^{n-1} |r_{i+1} - r_i|}{2\sum_{i=2}^{n} l_i(n - l_i)} \tag{1.19}$$

where r_i denotes the rank of $Y_{(i)}$, n is the sample size, and l_i is the number of indices j such that $Y_{(j)} \geq Y_{(i)}$. When there are no ties, this expression reduces to Equation (1.18).

It has been proved in Chatterjee (2021) that ξ_n is a consistent estimator of a certain measure of dependence between the random variables X and Y. If Y is not almost surely a

constant, then as $n \to \infty$, $\xi_n(X,Y)$ converges almost surely to the deterministic limit, say $\xi(X,Y)$, which can be viewed as the theoretical definition of the Chatterjee correlation coefficient and is given by

$$\xi(X,Y) := \frac{\int \mathrm{Var}\big(\mathbb{E}(1_{\{Y \geq t\}}|X)\big)\, dF(t)}{\int \mathrm{Var}(1_{\{Y \geq t\}})\, dF(t)} \tag{1.20}$$

where F denotes the law of Y. This limit lies in the interval $[0,1]$, equals 0 if and only if X and Y are independent, and equals 1 if and only if there exists a measurable function $f : \mathbb{R} \to \mathbb{R}$ such that $Y = f(X)$ almost surely.

The Chatterjee correlation coefficient is intentionally designed to be asymmetric with respect to X or with respect to Y. To assess whether X can be expressed as a function of Y, one should compute $\xi_n(Y,X)$ rather than $\xi_n(X,Y)$. If a symmetric measure of correlation is desired, a natural approach is to take the maximum of $\xi_n(X,Y)$ and $\xi_n(Y,X)$. It has been shown that this symmetric version converges in probability to $\max\{\xi(X,Y), \xi(Y,X)\}$. Extensions of this framework to multivariate dependence and conditional dependence have been proposed; see Azadkia and Chatterjee (2021) . It follows from the definition that $\xi(X,Y) \in [0,1]$ since $\mathrm{Var}(1_{\{Y \geq t\}}) \geq \mathrm{Var}(\mathbb{E}(1_{\{Y \geq t\}}|X))$ for every t. If X and Y are independent, then $\mathbb{E}(1_{\{Y \geq t\}}|X)$ is a constant, implying $\xi(X,Y) = 0$. Conversely, if Y is a measurable function of X, then $\mathbb{E}(1_{\{Y \geq t\}}|X) = 1_{\{Y \geq t\}}$, and hence $\xi(X,Y) = 1$. Although the limiting value of ξ_n lies in $[0,1]$, in practice the sample estimate may occasionally be negative. This occurs because the numerator in the definition (1.19) can exceed its expected range due to random fluctuations, and since the formula subtracts from 1, the resulting value can fall below zero.

In the empirical analysis that follows, we employ the Chatterjee correlation coefficient to quantify dependence among various cryptocurrencies and financial indices, using daily and weekly returns as well as standardized price data. The computations are

carried out using the `XICOR` package in R , which provides both estimate of ξ_n and corresponding p-values for testing the independence.

1.5 Pairs Trading Strategy and Cointegration Analysis

Pairs trading is a market-neutral strategy that capitalizes on the mean-reverting behavior of the price spread between two historically correlated financial assets. The effectiveness of this strategy hinges on the identification of asset pairs whose prices exhibit a long-term equilibrium relationship, despite short-term deviations. This equilibrium relationship is typically characterized through the concept of cointegration, which provides a formal statistical framework for modeling such comovements. In this section, we first present the theoretical foundation and practical implementation of the pairs trading strategy, followed by a detailed discussion of cointegration analysis as a diagnostic tool for identifying suitable trading pairs.

1.5.1 Pairs Trading Strategy

The pairs trading strategy leverages the concept of cointegration to exploit the relative mispricing between two assets, under the assumption that their prices will revert to a long-term equilibrium. The general idea is to buy the undervalued asset and sell the overvalued one. In general, we consider two assets from the same natural category and may have similar risk factors. Time series plots of their daily log prices can help assess whether they exhibit certain characteriestics of comovement. We then test for cointegration between the two series. For more details on pairs trading and staitistical arbitrage, see Vidyamurthy (2004) and Pole (2011).

Based on the arbitrage pricing theory (APT) in finance, if two stocks have similar characteristics, then the prices of both stocks must be more or less the same (See Bailey (2005)). Tsay (2010) presented the theoretical framework for corresponding trading

strategy. Let P_{it} denote the observed price of stock i at time t, and define the log price as $p_{it} = \ln(P_{it})$. If the two stocks have similar risk profiles, their log prices p_{1t} and p_{2t} are expected to follow a common trend and be cointegrated based on APT. That is, there must exist a linear combination

$$\omega_t = p_{1t} - \gamma p_{2t}$$

such that ω_t is unit-root stationary. This stationary series ω_t is referred to as the *spread* between the two log prices and it must then exhibit mean-reverting behavior.

Consider a portfolio that is long one dollar of stock 1 and short γ dollars of stock 2. The log return of this portfolio over a time interval i is given by

$$\begin{aligned} r_{p,t+i} &= (p_{1,t+i} - p_{1t}) - \gamma(p_{2,t+i} - p_{2t}) \\ &= (p_{1,t+i} - \gamma p_{2,t+i}) - (p_{1t} - \gamma p_{2t}) \\ &= \omega_{t+i} - \omega_t. \end{aligned} \tag{1.21}$$

Thus, the portfolio return depends on the change in the spread over time and is independent of the mean of spread. The pair-trading strategy involves exploiting deviations of the spread from its mean. Since ω_t is mean-reverting, we can initiate a trade when ω_t deviates significantly from its mean μ_ω, and close the position when it reverts. Let Δ denote a threshold for significant deviation. A simple trading rule is

- Enter a trade at time t if $\omega_t = \mu_\omega - \Delta$: Go long on stock 1 with one dollar and short γ dollars on stock 2.
- Exit the trade at time $t+i$ (where $i > 0$) when $\omega_{t+i} = \mu_\omega + \Delta$.

Typically, Δ is chosen to be close to or slightly larger than one standard deviation of the spread, ensuring a reasonable probability (the probability is about 30% under the normality assumption) of observing such deviations. These excursions between the lower

and upper thresholds represent trading opportunities. In practice, the strategy is implemented and evaluated on an out-of-sample period, and the choice of the Δ is based on several factors including trading costs, marginal interest rates, and bid–ask spreads of the two stocks.

1.5.2 Conintegration Analysis

To assess whether two stocks are suitable for pairs trading, a standard approach involves testing for cointegration between their log prices. Specifically, we consider the linear regression model

$$p_{1t} = \beta_0 + \beta_1 p_{2t} + \varepsilon_t,$$

where β_0 and β_1 are unknown regression parameters, and ε_t denotes the random error term. We assume that the errors are randomly and independently distributed as normal with $E(\varepsilon_t) = 0$ and constant variance $Var(\varepsilon_t) = \sigma^2$. The residual series r_t is computed as

$$r_t = p_{1t} - \hat{\beta}_0 - \hat{\beta}_1 p_{2t}$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are the least-squares estimates of the regression coefficients. We define the corresponding spread series as

$$\omega_t = p_{1t} - \hat{\beta}_1 p_{2t} = \hat{\beta}_0 + r_t,$$

As a sample surrogate of ω_t, If r_t and hence w_t is stationary, then p_{1t} and p_{2t} are said to be cointegrated. A preliminary diagnosis involves examining the time series plot of the residuals. If the series r_t fluctuates around a constant mean within a bounded range, this behavior may indicate the presence of stationarity.

To further investigate stationarity of a time series, we analyze its autocorrelation structure. The lag-ℓ autocorrelation of the series r_t, denoted by ρ_ℓ, is defined as the

correlation coefficient between r_t and $r_{t-\ell}$, which quantifies the linear dependence between r_t and its past value $r_{t-\ell}$. Under the assumption of weak stationarity, ρ_ℓ is a function depends only on the lag ℓ, and is given by

$$\rho_\ell = \frac{\mathrm{Cov}(r_t, r_{t-\ell})}{\sqrt{\mathrm{Var}(r_t)\,\mathrm{Var}(r_{t-\ell})}} = \frac{\mathrm{Cov}(r_t, r_{t-\ell})}{\mathrm{Var}(r_t)} = \frac{\gamma_\ell}{\gamma_0}.$$

By definition, we have $\rho_0 = 1$, $\rho_\ell = \rho_{-\ell}$, and $-1 \leq \rho_\ell \leq 1$. A weakly stationary process is uncorrelated if and only if $\rho_\ell = 0$ for all $\ell > 0$.

Given a sample of observations $\{r_t\}_{t=1}^T$, let $\bar{r} = \frac{1}{T}\sum_{t=1}^T r_t$ ($= 0$ for our case, by construction) denote the sample mean. The lag-ℓ sample autocorrelation coefficient is computed as

$$\hat{\rho}_\ell = \frac{\sum_{t=\ell+1}^T (r_t - \bar{r})(r_{t-\ell} - \bar{r})}{\sum_{t=1}^T (r_t - \bar{r})^2}, \qquad 0 \leq \ell < T-1.$$

If the sample autocorrelation functions (ACF) of r_t, i.e., the sequence $\hat{\rho}_1, \hat{\rho}_2, \hat{\rho}_3, \ldots$, decays exponentially, this provides an empirical support for the stationarity of r_t. This pattern of decay suggests that the influence of past values diminishes rapidly over time. In particular, for the first order autoregressive models such as AR(1),

$$r_t = \phi r_{t-1} + \varepsilon_t, \quad |\phi| < 1,$$

where ε_t is white noise. The theoretical autocorrelation function (ACF) for this process is then given by

$$\rho_\ell = \phi^\ell,$$

which decays exponentially as $\ell \to \infty$. Conversely, for the non-stationary processes, such as random walks (when $\phi = 1$),

$$r_t = r_{t-1} + \varepsilon_t,$$

, the ACFs remain close to 1 even for large lags and tends to decay very slowly (or not at all). Therefore, observing exponential decay in the sample ACF serves as a practical diagnostic for identifying stationarity in a residual series.

To formally test for stationarity, we consider the autoregressive model

$$r_t = \phi_1 r_{t-1} + \varepsilon_t, \tag{1.22}$$

where ε_t is a white noise process. The alternative hypothesis $H_a : \phi_1 < 1$ implies the stationarity of the series, while the null hypothesis $H_0 : \phi_1 = 1$ corresponds to the presence of a unit root and hence nonstationarity. The least-squares (LS) estimate of ϕ_1 is given by

$$\hat{\phi}_1 = \frac{\sum_{t=1}^{T} r_{t-1} r_t}{\sum_{t=1}^{T} r_{t-1}^2}, \tag{1.23}$$

$$\hat{\sigma}_e^2 = \frac{\sum_{t=1}^{T} (r_t - \hat{\phi}_1 r_{t-1})^2}{T-1}. \tag{1.24}$$

where $r_0 = 0$ and T denotes the length of the series on sample size. The test statistic for the above null hypothesis of nonstationarity is given by

$$\text{DF-test} \equiv t \text{ ratio} = \frac{\hat{\phi}_1 - 1}{\hat{\text{std}}(\hat{\phi}_1)} = \frac{\sum_{t=1}^{T} r_{t-1} \varepsilon_t}{\hat{\sigma}_e \sqrt{\sum_{t=1}^{T} r_{t-1}^2}}.$$

and is referred to as the Dickey–Fuller (DF) test. See Dickey and Fuller (1979) , Chan and Wei (1988) and Phillips (1987) for further details. Null is rejected in favor of alternative hypothesis if the computed P value is small.

For more general situations, in the econometric literature, autoregressive models of order p, denoted AR(p), are frequently employed. Let r_t denote the time series of interest. To test the existence of a unit root and thus nonstationarity in an AR(p) process, one may consider the regression model

$$r_t = c_t + \beta r_{t-1} + \sum_{i=1}^{p} \phi_i \Delta r_{t-i} + e_t,$$

where $\Delta r_j = r_j - r_{j-1}$ is the first-differenced series, and c_t is a deterministic component that may be zero, a constant, or a linear trend (e.g., $c_t = \omega_0 + \omega_1 t$).

Here also the null hypothesis $H_0 : \beta = 1$ (nonstationarity) is tested against the alternative $H_a : \beta < 1$. The test statistic is

$$t^* = \frac{\hat{\beta} - 1}{\hat{\text{std}}(\hat{\beta})},$$

where $\hat{\beta}$ is the least-squares estimate of β. The test statistic follows a t-distribution under H_0. This is known as the augmented Dickey–Fuller (ADF) test for stationary. See Dickey and Fuller (1979) . We reject the null hypothesis of nonstationarity if P value is small. We also employ this more general model and corresponding test statistic to test the cointegration.

1.6 Concluding Remarks

This chapter provided a brief review of the statistical methodologies employed in the analyses in this dissertation. More detailed discussions of these techniques can be found in references given earlier in the respective sections of this chapter. We must point out that the list of techniques used here to deal with similar problems is not exhaustive and many other alternative techniques can be applied to tackle these and other similar problems. Which of these techniques maybe more suitable is something that remains to be investigated. However, we feel that collectively, these techniques establish a relatively comprehensive framework for analyzing the behaviors of cryptocurrencies, energy supply and related predictive problems. The subsequent chapters build upon this foundation to deliver the empirical insights and practical applications.

CHAPTER TWO

CRYPTOCURRENCIES: BASIC DESCRIPTIVE ANALYSES

2.1 Introduction

Cryptocurrencies, first introduced in the late 2000s, have revolutionized the global financial landscape, emerging as a significant player in modern finance markets. Unlike conventional financial instruments, the cryptocurrency is a digital currency designed to work as a medium of exchange through a computer network without reliance on any central authority such as a government or bank, to uphold or maintain it. This decentralized nature is discussed in depth by Hägele (2024) and Milutinović (2018) . This architecture enables peer-to-peer transactions without any need for any centralized intermediaries.

The research about cryptocurrencies has increased rapidly. Blau (2017) studied the Bitcoin returns and found that Bitcoin's price movements are largely driven by speculative trading rather than any fundamental value. Cheah and Fry (2015) considered investigation into speculative bubbles in Bitcoin using econophysics-based models and found that the Bitcoin prices contain a substantial speculative component. Wątorek et al. (2021) investigated the structural complexity of cryptocurrency markets using multiscale analysis techniques. The study revealed that the crypto market is characterized by high volatility, non-linearity, and a high degree of interconnectedness with traditional financial systems, indicating both opportunities and risks for investors. ElBahrawy et al. (2017) analyzed the behavior of 1,469 cryptocurrencies introduced between 2013 and 2017, revealing that although many new coins emerge and vanish rapidly, their key statistical properties, such as market share distribution and turnover, remain stable over time. Liu and Serletis (2019) used GARCH-in-mean models to analyze the relationship between volatility and returns

in major cryptocurrencies. It suggest that cryptocurrencies are increasingly integrated with global financial systems and that their volatility has broader implications for risk management and portfolio diversification.

This chapter thus aims to provide a foundational statistical and econometric overview of the behaviors of a few leading cryptocurrencies, focusing on their return distributions, volatility patterns, and interdependencies. The analysis begins with a comprehensive examination of daily return distributions for seven representative prominent cryptocurrencies: Bitcoin (BTC), Ethereum (ETH), Ethereum Classic (ETC), Tether (USDT), Cardano (ADA), XRP (XRP), and Dogecoin (DOGE). Why only seven currencies? It is because crypto assets are fairly new for the market and in order to do a fair and meaningful comparison we must have enough data. Many cryptocurrencies have appeared only lately and several currencies in the past have also vanished soon after their appearance. Further, these currencies must have enough liquidity. Thus we consider the data, obtained from Yahoo Finance from December 31, 2017, to June 17, 2024, which includes daily open, close, low, and high prices of above cryptocurrencies. These assets were selected based on consideration of their market capitalization, trading volume, longevity, and relevance in both retail and institutional portfolios. Through descriptive statistics, visual diagnostics, and normality tests, we assess the distributional properties of these assets, highlighting features such as skewness and the presence of outliers.

The chapter explores the interrelationships among these cryptocurrencies using various association measures, such as Pearson, Spearman, Kendall, and the more recently introduced Chatterjee correlation. These metrics may provide insights into both linear as well as nonlinear dependencies, which are crucial for portfolio diversification and risk management. The analysis in this chapter serves as a statistical foundation for the more

advanced multivariate and predictive modeling techniques developed in subsequent chapters.

We conduct formal tests for normality using the Shapiro–Wilk and Kolmogorov–Smirnov tests, as well as visual assessments by using Q–Q plots. Through this multifaceted approach, the chapter aims to provide a robust statistical foundation for understanding cryptocurrency behavior, offering valuable insights for investors, policymakers, and researchers interested in the dynamics of digital asset markets.

2.2 Summary Statistics, Outliers, Influential Observations

As indicated earlier, using a comprehensive dataset spanning from December 31, 2017, to June 17, 2024, we analyze seven major cryptocurrencies: Bitcoin, Ethereum, Ethereum Classic, Tether, Cardano, XRP, and Dogecoin. For each cryptocurrency, the dataset includes 2361 daily observations, with each entry containing the open price, closing price, lowest price, and highest price recorded on that day. These data were retrieved from Yahoo Finance, providing a consistent and reliable source for historical market activity. To quantify daily performance, we compute the daily return y_t for each cryptocurrency using the formula $y_t = \frac{x_t - x_{t-1}}{x_{t-1}}$, where x_t denotes the closing price on day t, and x_{t-1} is the closing price on the previous trading day. Since the Cryptos are traded all twenty-four hours, we believe that closing prices are the prices at the end of the day. This return metric captures the relative change in value from one day to the next and serves as the primary variable for subsequent statistical analyses. In order to analyze trends over a weekly period, we use the seven-day moving average of the daily returns given by

$$u_t = \frac{y_{t-6} + y_{t-5} + y_{t-4} + y_{t-3} + y_{t-2} + y_{t-1} + y_t}{7} \tag{2.1}$$

This seven day equally weighted average helps smooth out short-term fluctuations and allow us to highlight any longer-term trends in the daily returns. Similarly, the seven-day moving average of the daily closing prices

$$w_t = \frac{x_{t-6} + x_{t-5} + x_{t-4} + x_{t-3} + x_{t-2} + x_{t-1} + x_t}{7} \tag{2.2}$$

This measure provides a smoothed view of price movements, allowing for better analysis of trends without the noise of the daily volatility. The summary statistics for the daily returns are shown in Table 2.1.

Table 2.1: Descriptive Statistics for Various Cryptocurrencies

	Mean	Std.Dev	Median	Min	Max	25th	75th	Skew	Kurtosis
Bitcoin	0.00131	0.0359	0.000810	-0.372	0.187	-0.0138	0.0163	-0.346	10.8
Cardano	0.00124	0.0549	-0.000369	-0.396	0.380	-0.0264	0.0246	0.579	8.29
Dogecoin	0.00405	0.0995	-0.000811	-0.403	3.56	-0.0230	0.0193	20.3	699.0
Ethereum	0.00172	0.0459	0.000676	-0.423	0.259	-0.0187	0.0226	-0.274	9.23
Ethereum Classic	0.00157	0.0577	0.0000584	-0.397	0.423	-0.0226	0.0228	0.856	12.1
Tether	0.000000685	0.00347	-0.00000700	-0.0512	0.0548	-0.000437	0.000453	0.672	59.0
XRP	0.000911	0.0569	-0.000975	-0.423	0.731	-0.0209	0.0194	1.98	27.5

The Figure 2.1 presents boxplots for each currency, providing an overview of their respective distributions. These boxplots include labels indicating the maximum value for each currency.

There were some striking features. The maximum daily return occurred for Dogecoin on January 28, 2021, with a value of 355.55%. This increase was attributed to a tweet comment from Tesla CEO Elon Musk about Dogecoin. Trading volumes were also significantly larger around this date. Table 2.2 exhibits the price data surrounding the

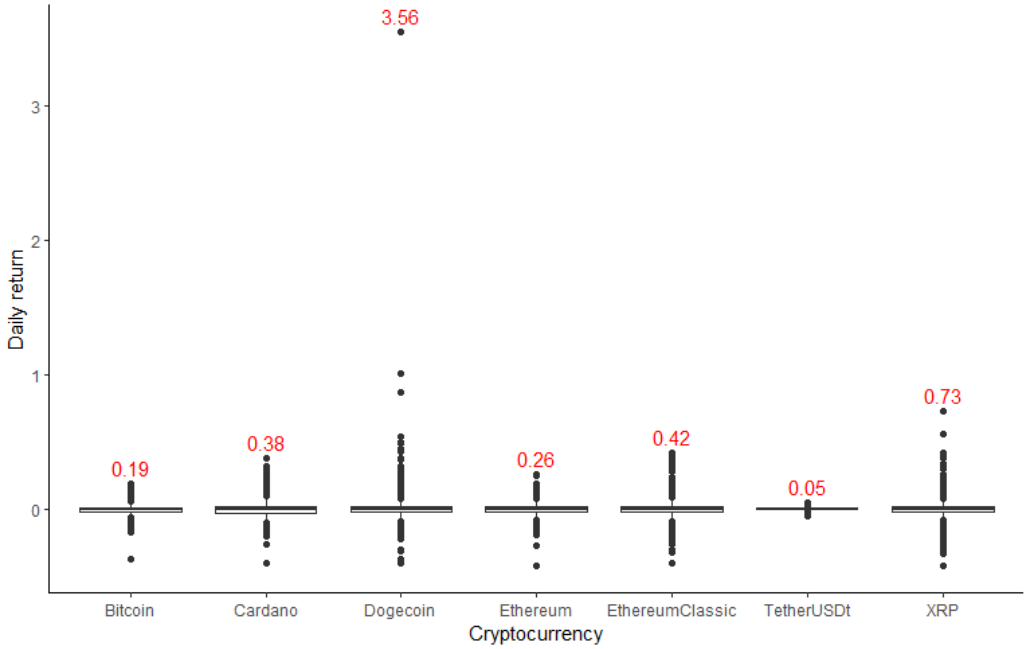


Figure 2.1: Boxplots of Cryptocurrencies Daily Return Distribution

significant increase in Dogecoin's price. Tether exhibits a relatively narrow range of daily returns. After removing the maximum value of daily returns for Dogecoin, a revised boxplot is shown in Figure 2.2.

Table 2.2: Price Data for Dogecoin Around the Date of Maximum Daily Return

Date	Open	High	Low	Close	Volume	Daily Return	MAvg Return
2021-01-25	0.008727	0.008879	0.008276	0.008383	180820875	-0.039418	-0.011812
2021-01-26	0.008382	0.008456	0.008022	0.008255	161173749	-0.015269	-0.012856
2021-01-27	0.008255	0.008259	0.007294	0.007482	204797186	-0.093640	-0.025637
2021-01-28	0.007481	0.034177	0.007351	0.034084	10971544561	3.555466	0.496383
2021-01-29	0.043734	0.077973	0.032341	0.047162	25403310432	0.383699	0.544935
2021-01-30	0.046803	0.049901	0.022488	0.028176	8735576553	-0.402570	0.486503

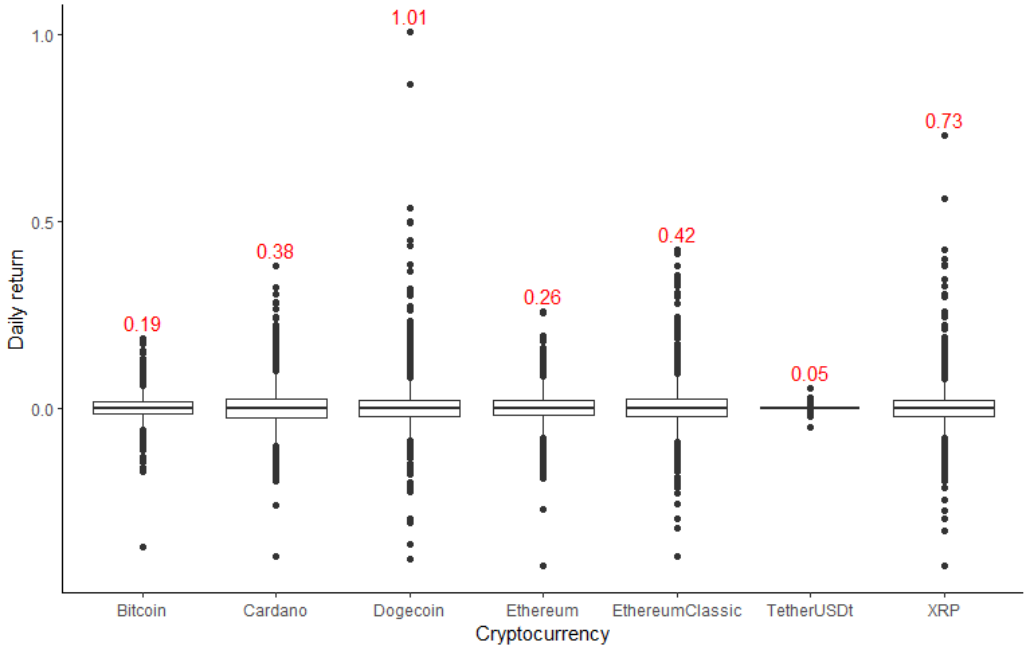


Figure 2.2: Boxplots of Cryptocurrencies Daily Return Distribution (Removing Max Value of Dogecoin)

From the boxplots in Figure 2.1 and 2.2, it is evident that the daily returns of Tether are highly concentrated around the median value. There is simply not much variation in daily returns for Tether. To gain further insights into the distributions of each cryptocurrency, we construct histograms of daily returns. In Figure 2.3, the histograms of the daily returns for all the cryptocurrencies are presented. It can be observed that the daily returns of all cryptocurrencies except Tether appear to have symmetric distributions. Tether's returns are notably centered around zero.

To analyze the time series characteristics of each cryptocurrency, individual histograms of daily returns and plots of seven-day moving averages of closing prices as well as daily returns over time are presented in Figures 2.4 - 2.24.

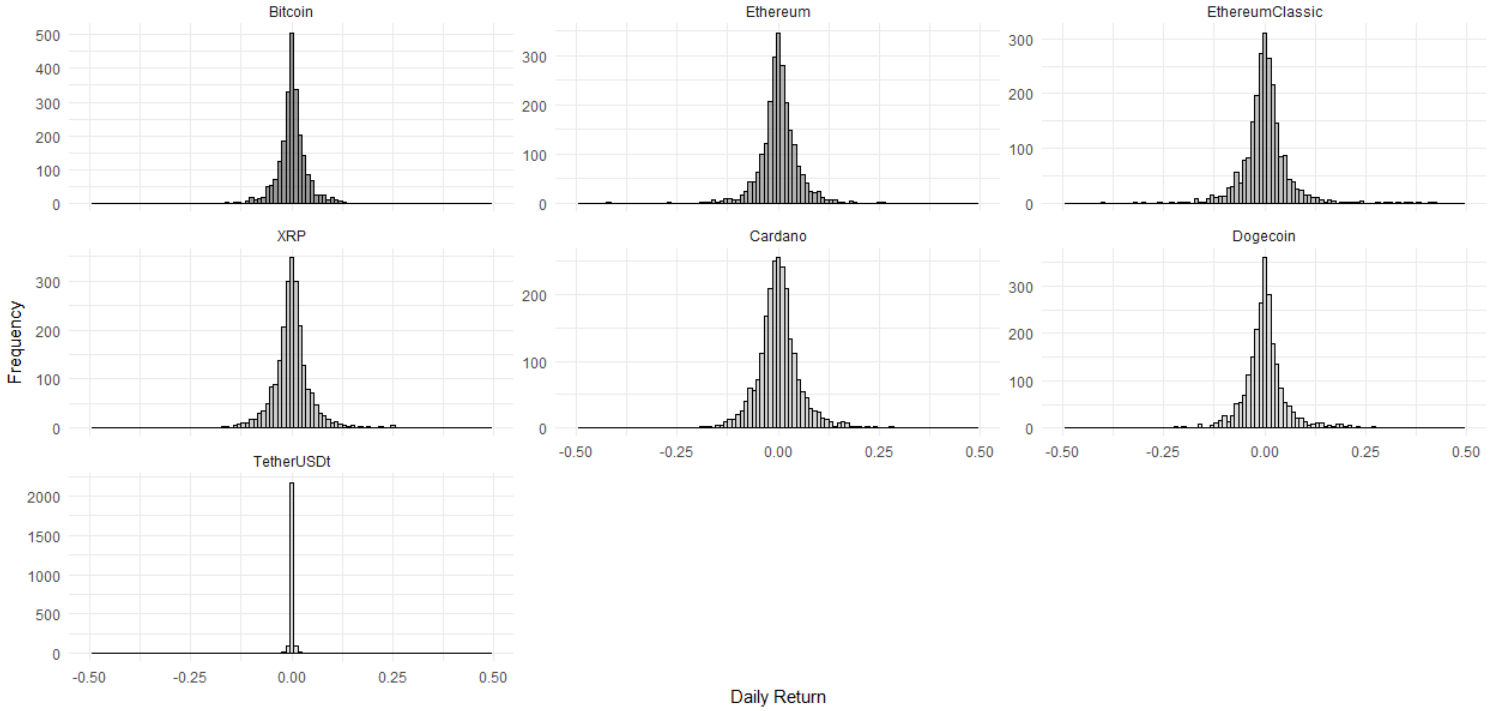


Figure 2.3: Histogram of Daily Returns for Selected Cryptocurrencies (Note: Y-axis scales are not directly comparable across cryptocurrencies due to differing return distributions.

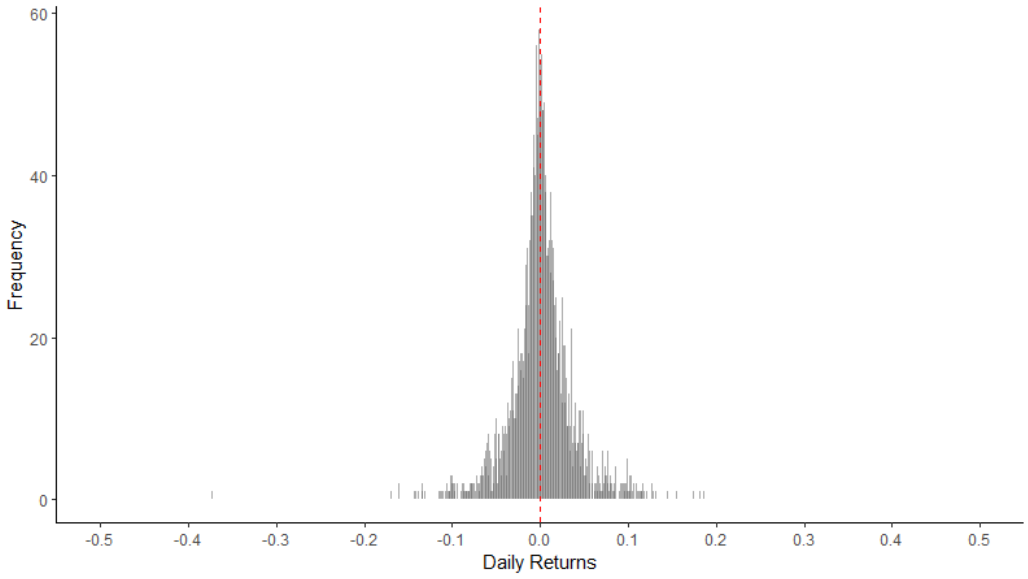


Figure 2.4: Distribution of Daily Return: Bitcoin

- When considering the seven-day moving average returns, Ethereum exhibits greater fluctuations compared to Bitcoin. Bitcoin and Ethereum both experienced a significant increase in its seven-day moving average prices around 2022, followed by a decline in 2023. A rising trend for prices is exhibited from the beginning of 2024.

- Based on the same plots, one observes that the Ethereum Classic witnessed a substantial increase in 2021, followed by a long-term decreasing trend. Additionally, it also demonstrates a greater degree of fluctuations in returns compared to Ethereum.

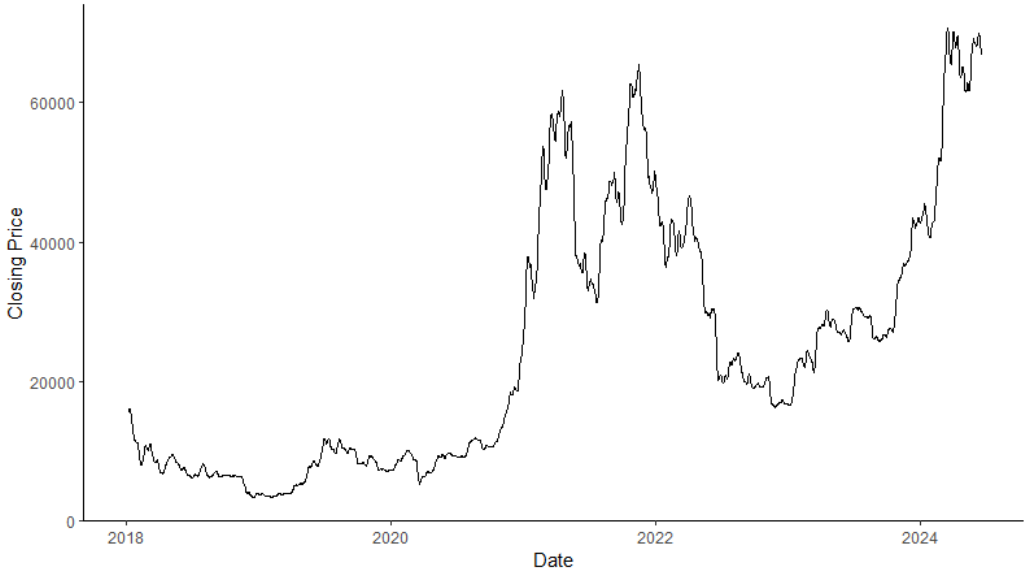


Figure 2.5: Seven-day moving average of Prices: Bitcoin

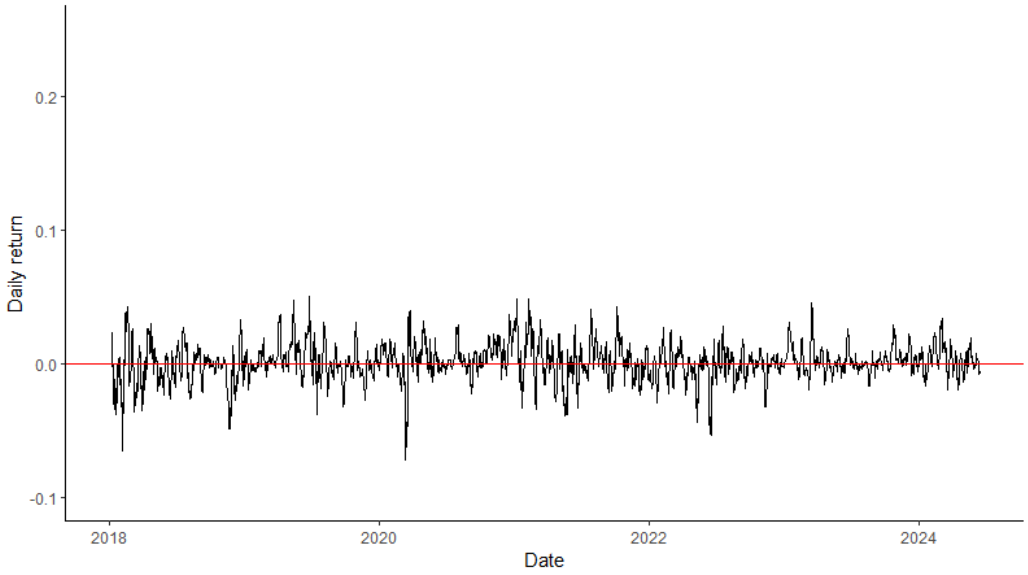


Figure 2.6: Seven-day moving average of Daily Returns: Bitcoin

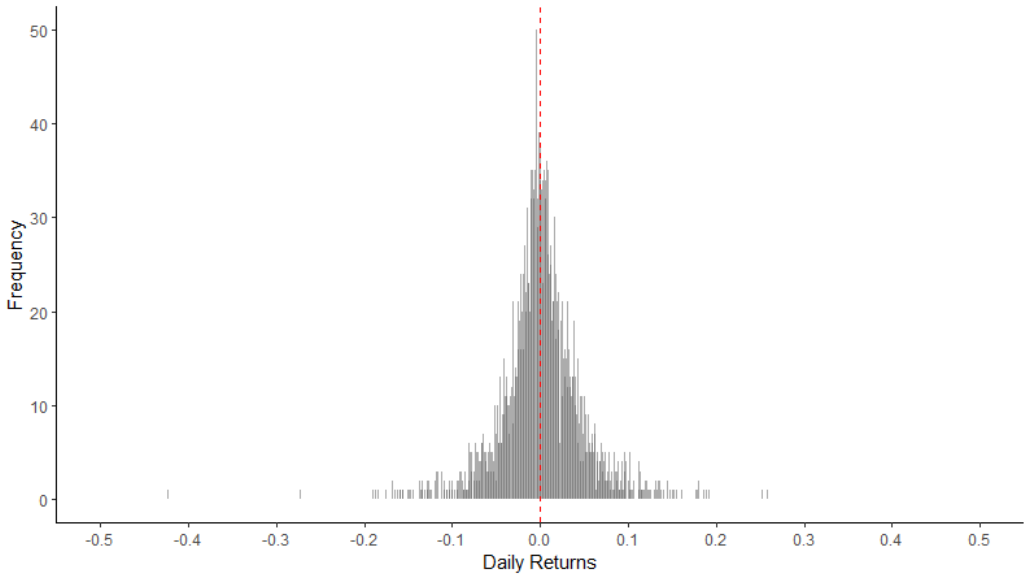


Figure 2.7: Distribution of Daily Return: Ethereum

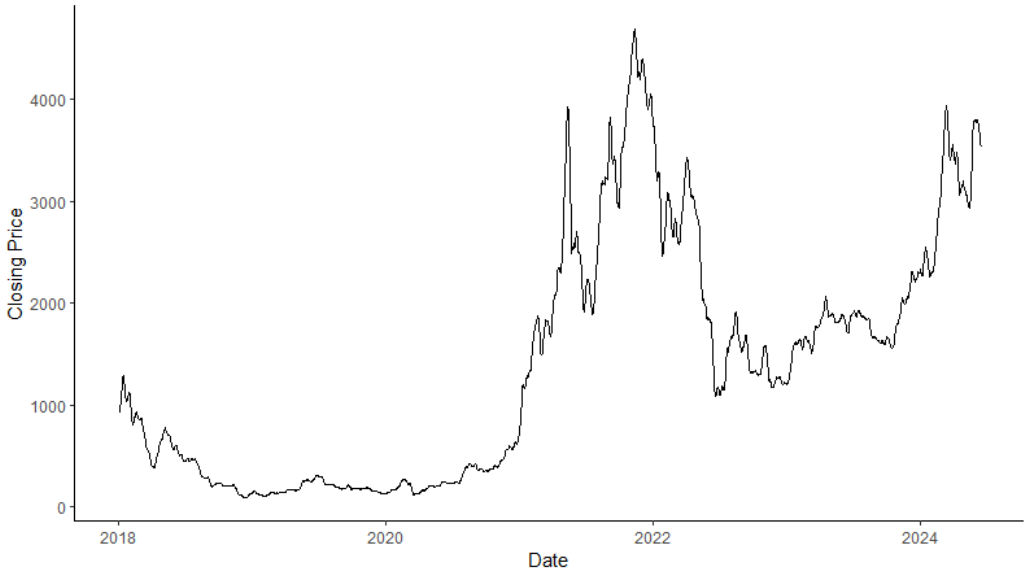


Figure 2.8: Seven-day moving average of Prices: Ethereum

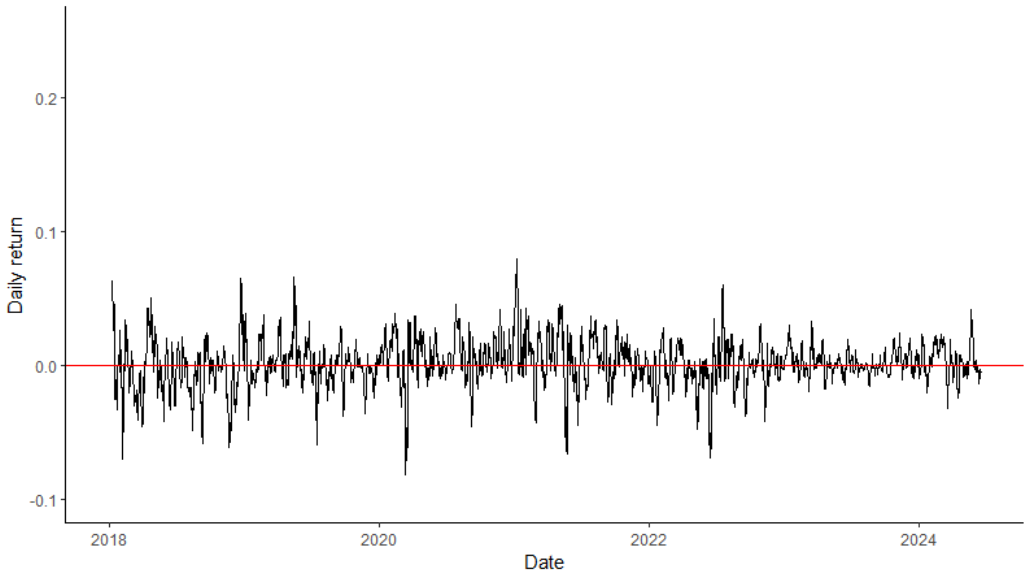


Figure 2.9: Seven-day moving average of Daily Returns: Ethereum

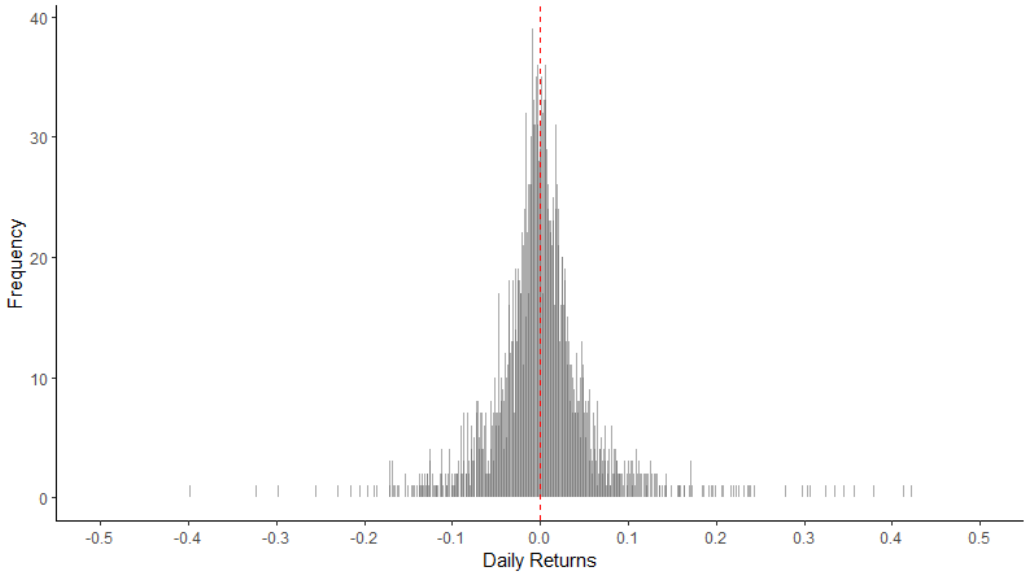


Figure 2.10: Distribution of Daily Return: Ethereum Classic

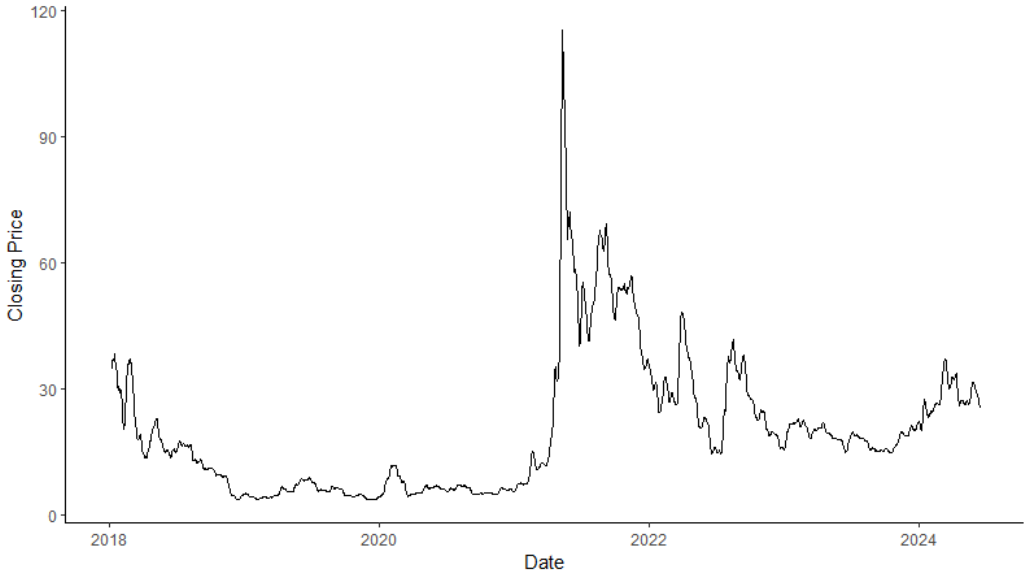


Figure 2.11: Seven-day moving average of Prices: Ethereum Classic

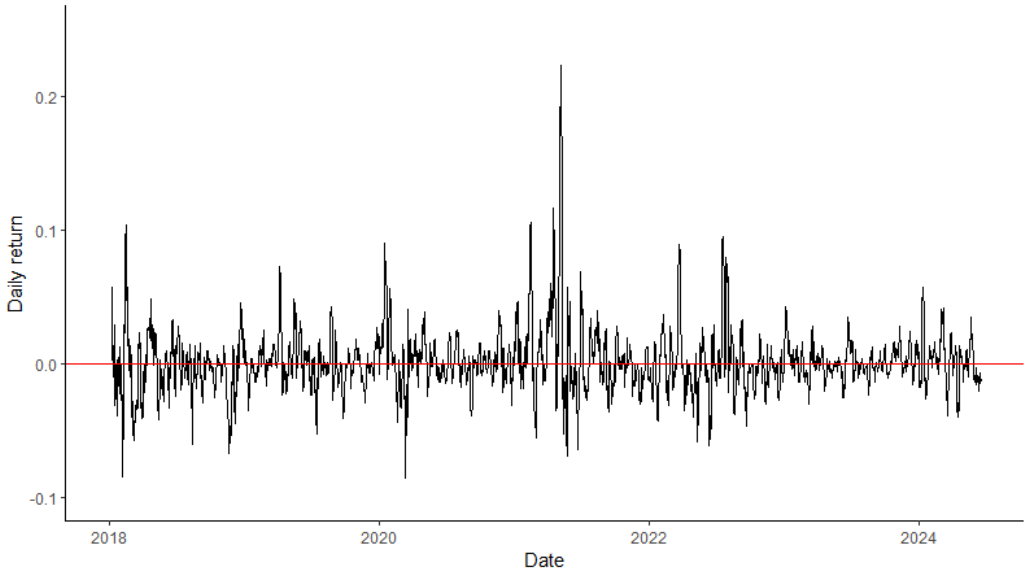


Figure 2.12: Seven-day moving average of Daily Returns: Ethereum Classic

- Cardano exhibits a similar trend as in Ethereum Classic in terms of moving average of prices. It reached its maximum price (during the study window) in 2021 and has since experienced a decreasing trend.

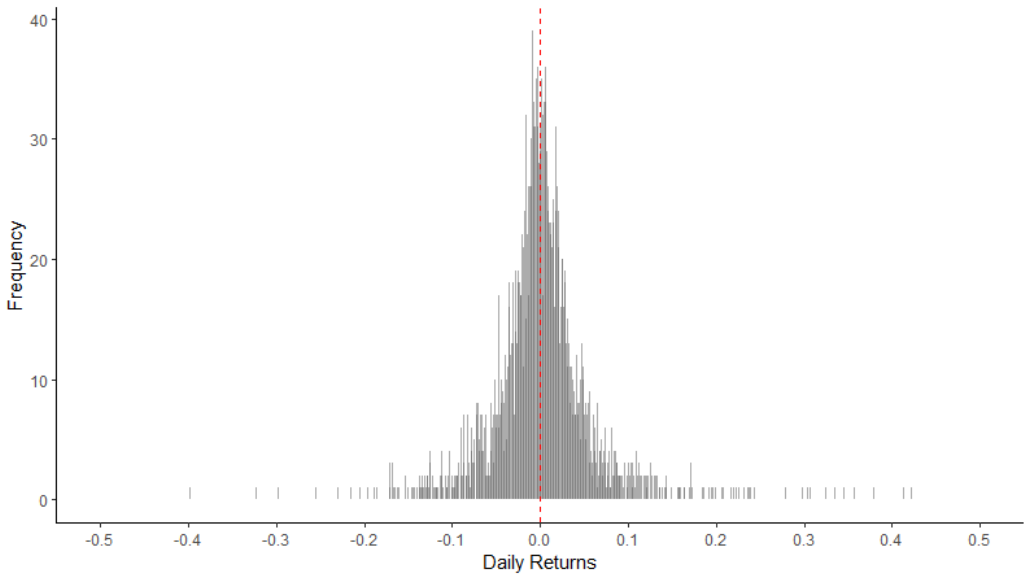


Figure 2.13: Distribution of Daily Return: Cardano

- Tether stands out from all other coins with a distinct distribution pattern. It maintains a consistently stable moving average of returns around zero. In terms of the moving average of prices, it experienced relatively significant fluctuations before 2020, but has since stabilized around a value of one.

- XRP demonstrates a notable decrease in its moving average of prices at the start of 2018, and then in 2021 there was a subsequent increase leading to a peak value. Since 2023, there has been a gradual upward trend in the moving average of prices.

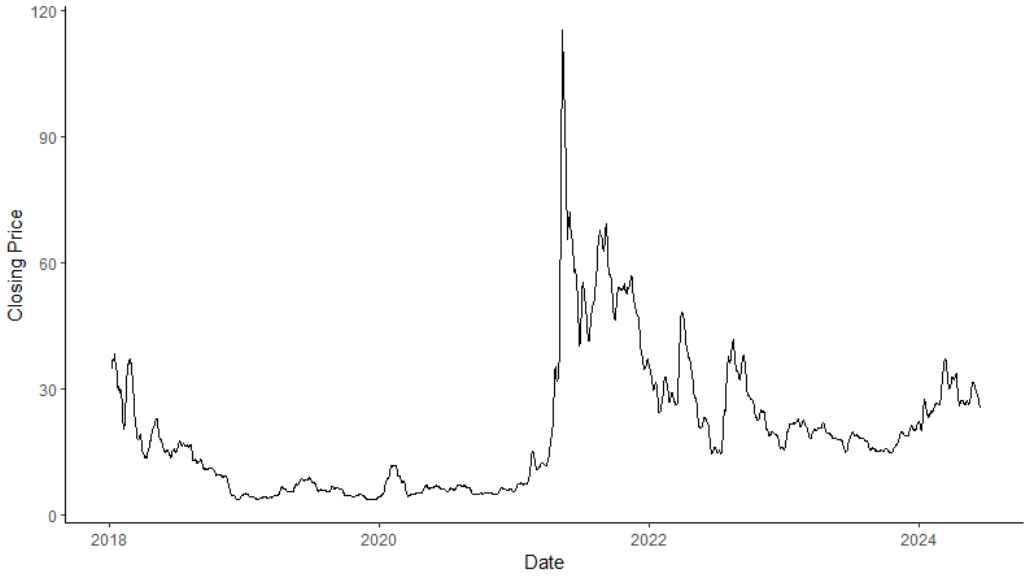


Figure 2.14: Seven-day moving average of Prices: Cardano

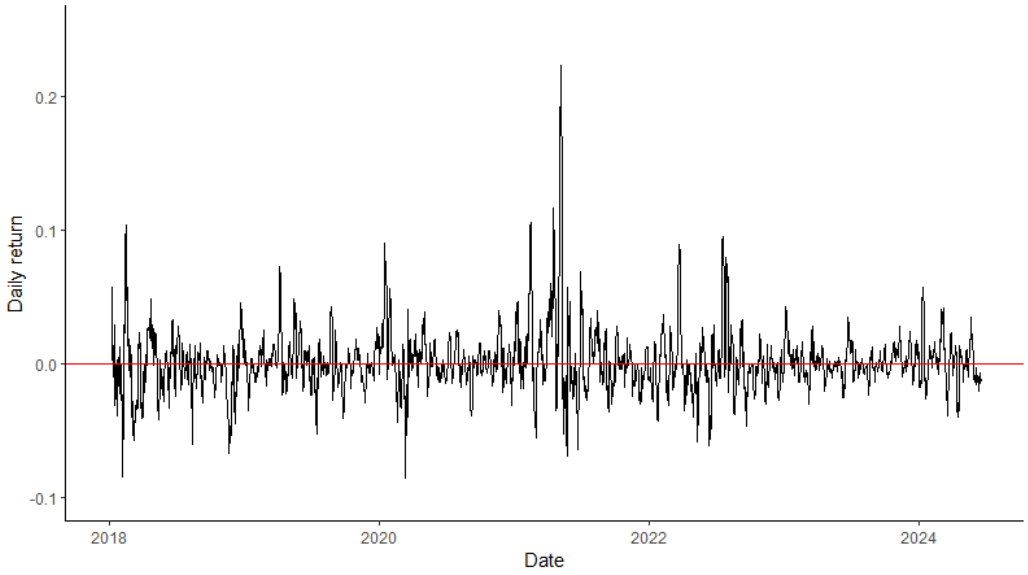


Figure 2.15: Seven-day moving average of Daily Returns: Cardano

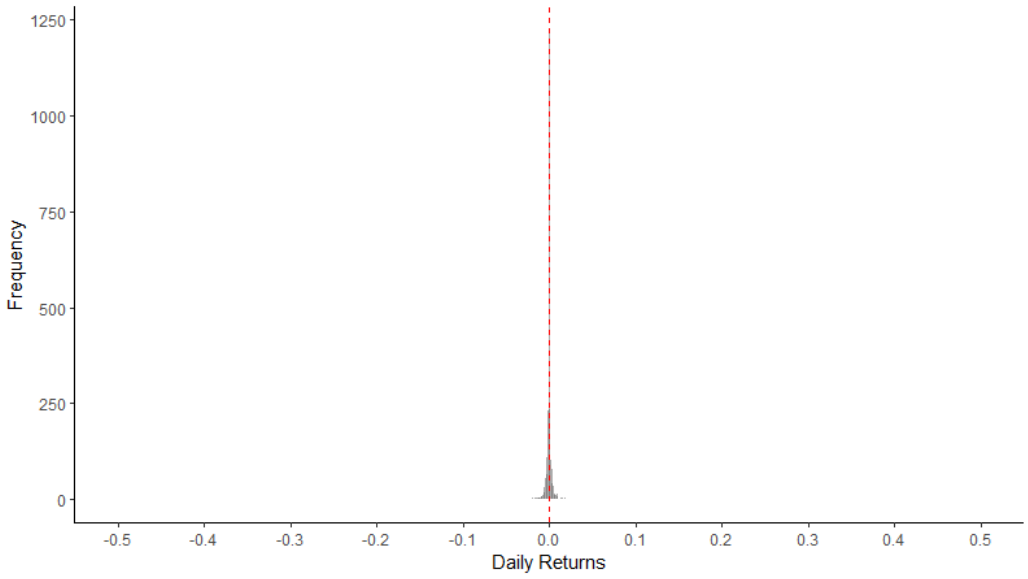


Figure 2.16: Distribution of Daily Return: Tether

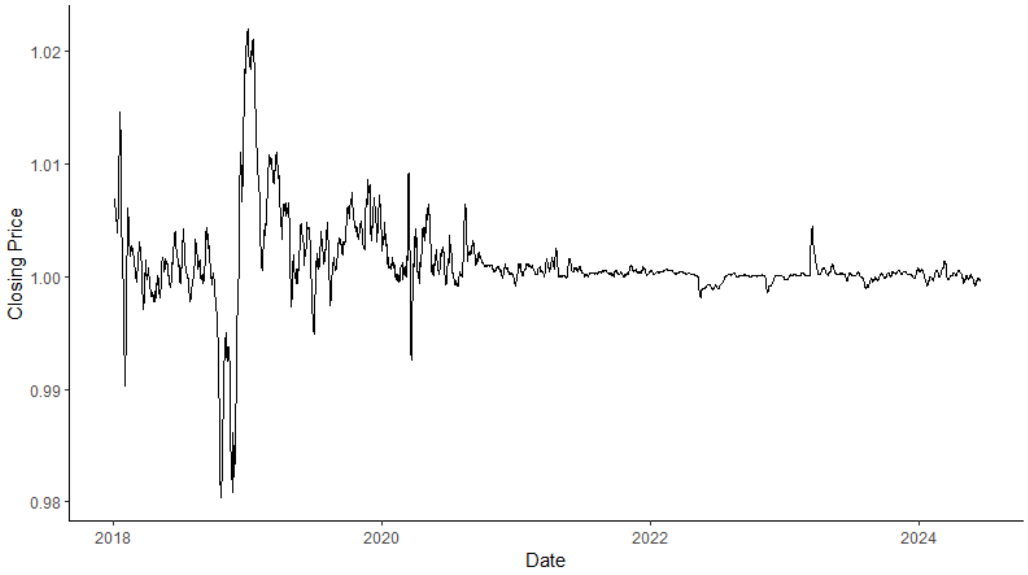


Figure 2.17: Seven-day moving average of Prices: Tether

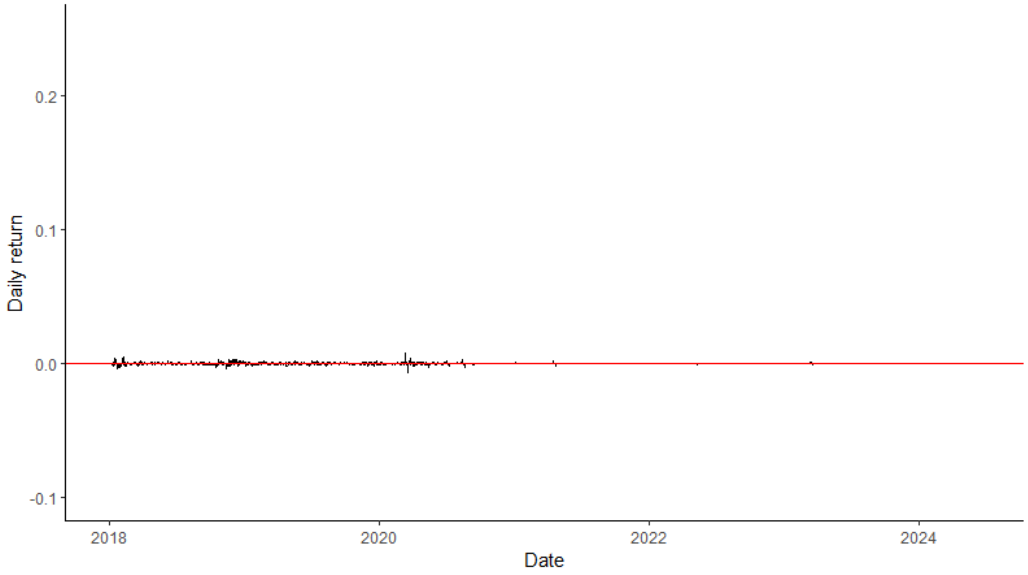


Figure 2.18: Seven-day moving average of Daily Returns: Tether

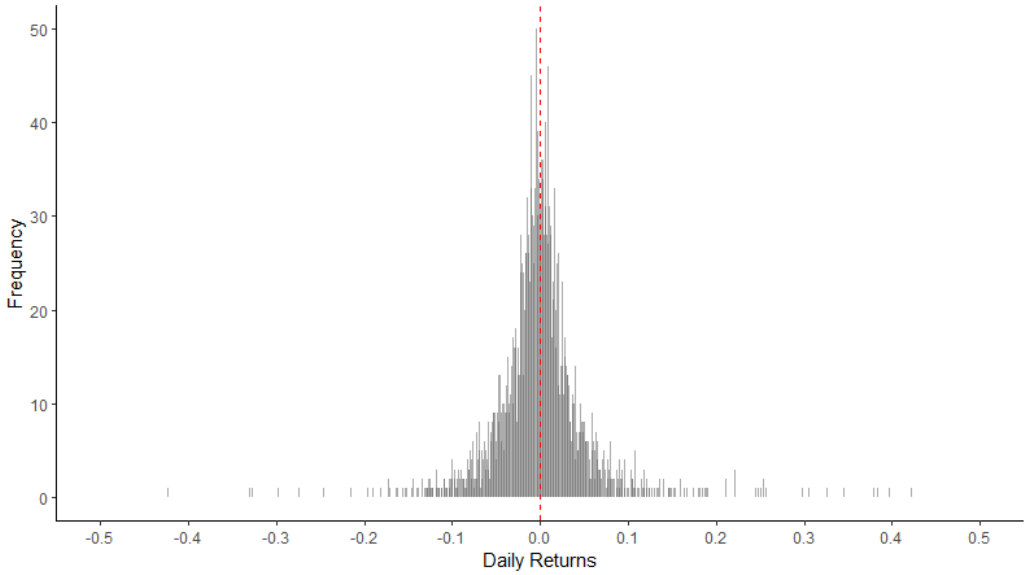


Figure 2.19: Distribution of Daily Return: XRP

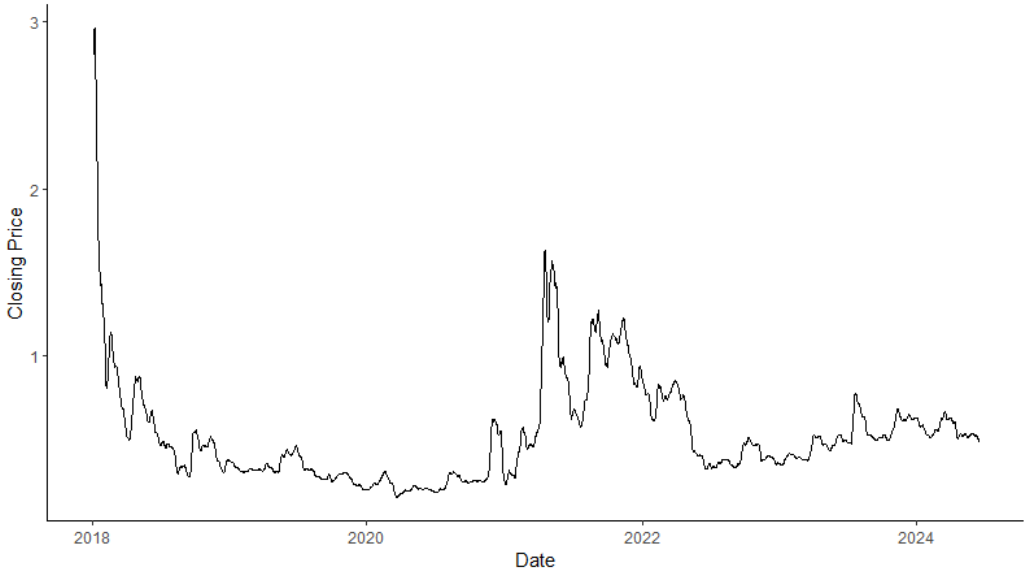


Figure 2.20: Seven-day moving average of Prices: XRP

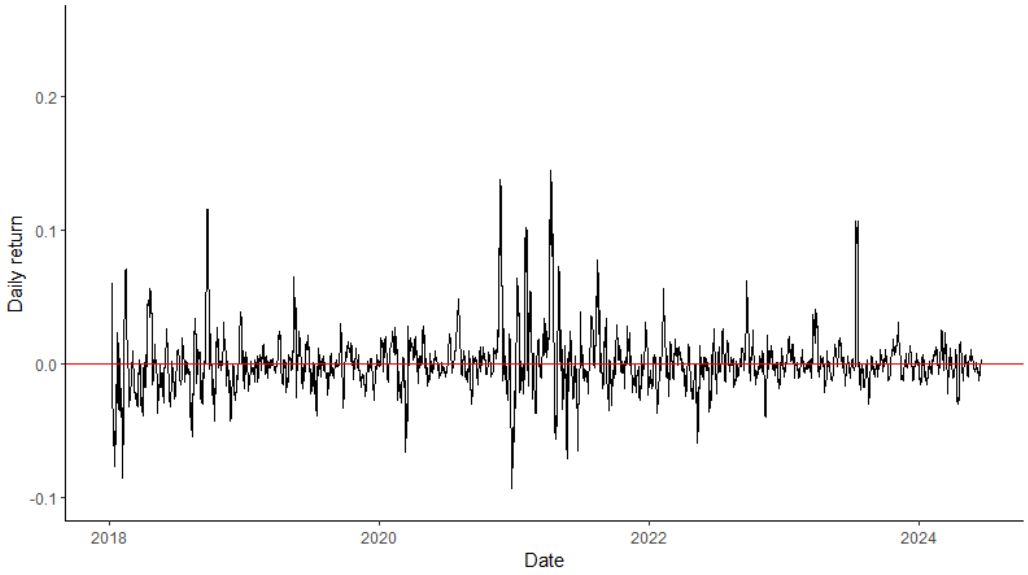


Figure 2.21: Seven-day moving average of Daily Returns: XRP

The moving average of returns of XRP exhibit significant fluctuations in 2021, followed by a more stable pattern in subsequent years, except for a sudden peak sometime in late 2023.

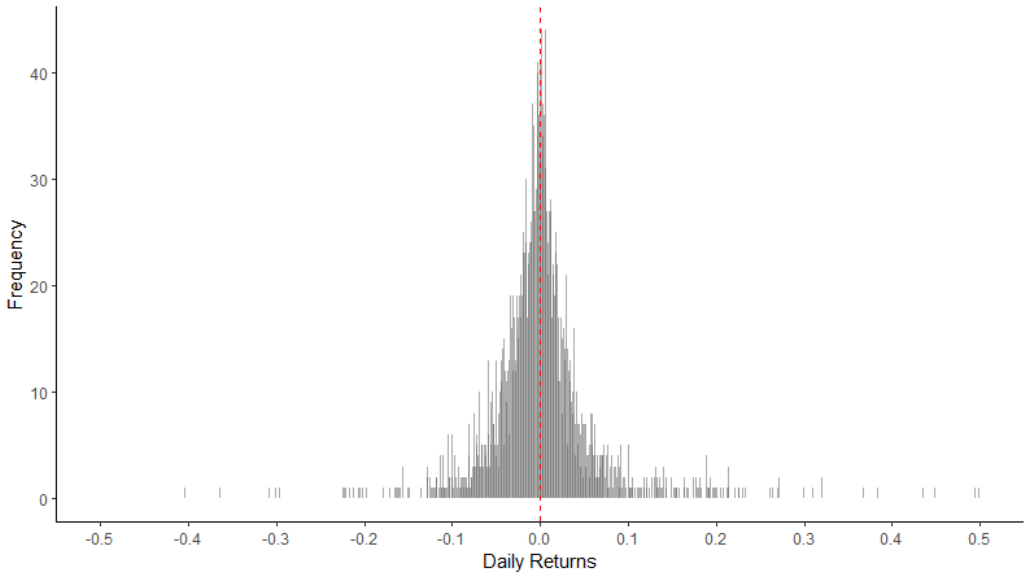


Figure 2.22: Distribution of Daily Return: Dogecoin

- Before 2021, Dogecoin exhibited a relatively stable moving average of prices. However, in 2021, there was a substantial surge in its value, followed by a long-term declining trend that persisted until 2024. In terms of the moving average of returns, Dogecoin experienced much greater fluctuations compared to other cryptocurrencies.

For a better comparison of the time-series of the seven-day moving average of prices, we standardize the corresponding closing prices by

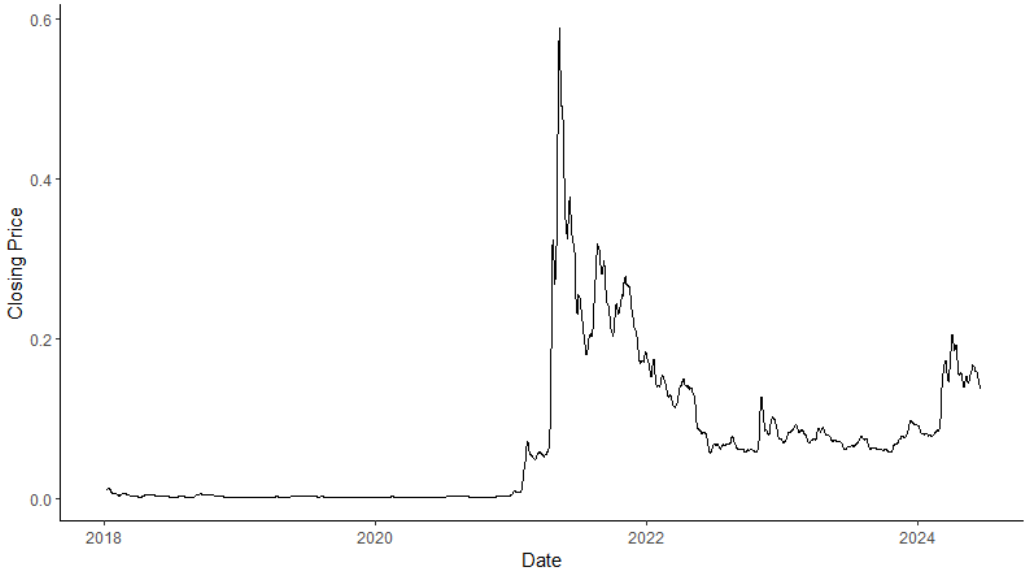


Figure 2.23: Seven-day moving average of Prices: Dogecoin

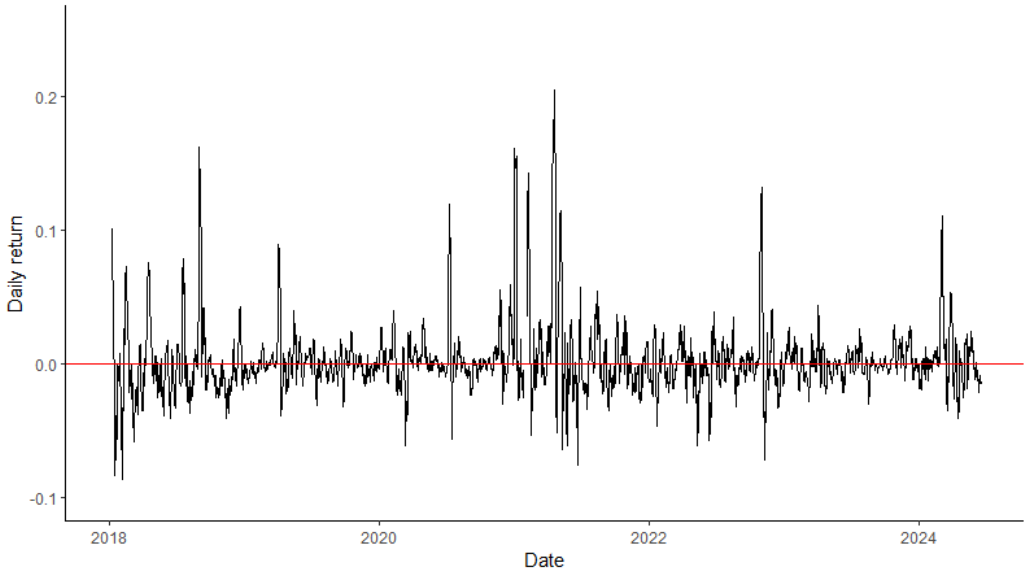


Figure 2.24: Seven-day moving average of Daily Returns: Dogecoin

$$S_t = \frac{x_t}{x_0} \times 100 \tag{2.3}$$

where x_t is the closing price on day t and x_0 is the closing price on Day 0, which is 2017-12-31 in this analysis. Clearly on Day 0, the standardized closing price is 100 for every currency.

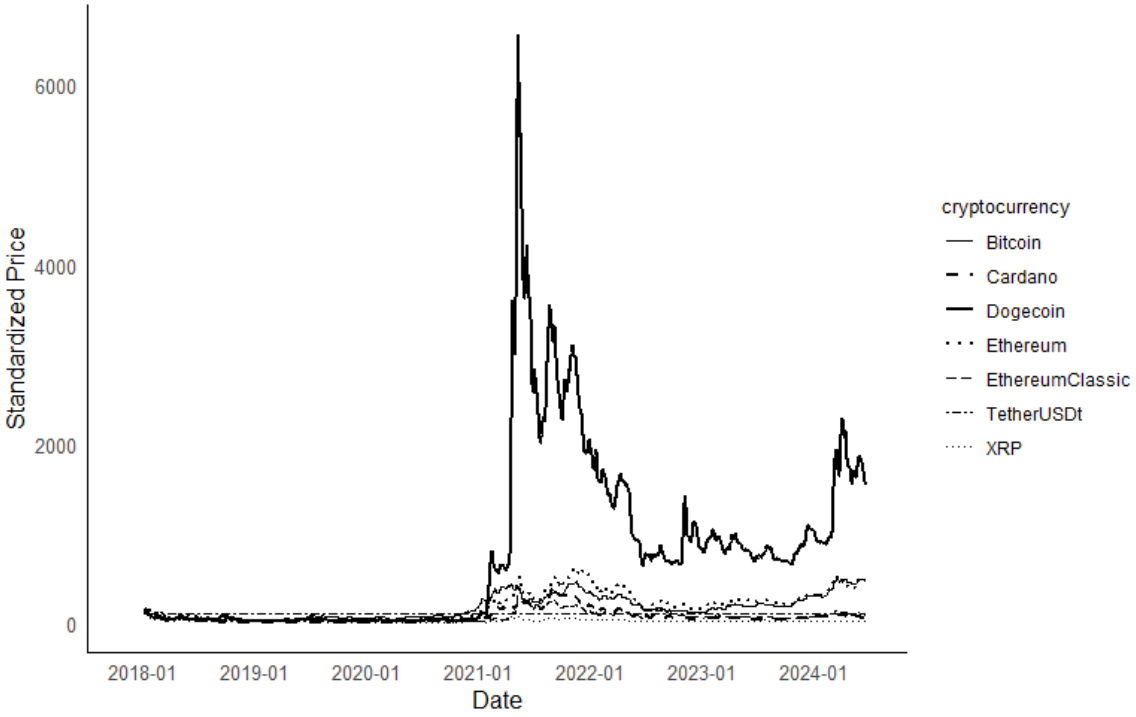


Figure 2.25: Seven-day moving average Standardized Price by Date

From the Figure 2.25 where the standardized prices are plotted, it is evident that Dogecoin experienced a significantly larger increase in 2021 compared to other coins. Additionally, Dogecoin has maintained a higher standardized prices thereafter. But much

of the gain was due to a sudden one day increase as indicated earlier. Figure 2.26 excludes the Dogecoin from the plot and this allows for a clearer visualization of the trends observed for the rest of the currencies. Tether exhibits a stable price throughout the entire study window. On the other hand, XRP shows a relatively lower standardized price following its decrease from the beginning of 2018. The remaining four coins (Bitcoin, Ethereum, Ethereum Classic, and Cardano) display a similar moving trend prior to January 2023. However, Bitcoin and Ethereum subsequently experience a greater increase compared to Ethereum Classic and Cardano.

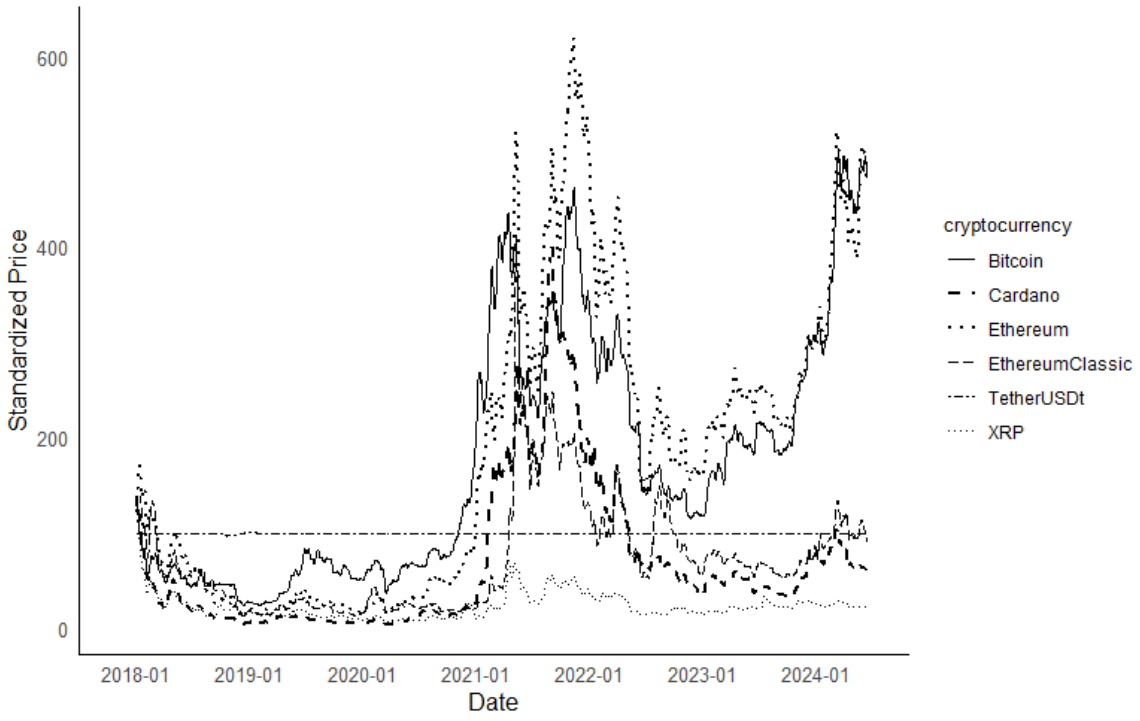


Figure 2.26: Seven-day moving average Standardized Price by Date (Excluding Dogecoin)

To study the trend after the price spikes in 2021 and 2022, we plot in Figure 2.27 the standardized prices by date, with 2022-12-31 serving as starting date. As earlier, the closing price of each cryptocurrency on the starting date ('Day 0') was taken as 100. Table 2.3 presents the actual closing prices of these cryptocurrencies on December 31, 2022.

In general, since the beginning of 2023, these standardized prices of cryptocurrencies appear to increase. Bitcoin shows a very substantial price growth, particularly noticeable around mid-2024, reaching the highest standardized price among the listed cryptocurrencies. Both Ethereum and Ethereum Classic exhibit growth trends, with Ethereum showing a more pronounced increase in standardized price compared to Ethereum Classic, especially towards mid-2024. Cardano and Dogecoin display moderate fluctuations, with notable upward movements around early 2024. XRP demonstrates stability with minimal fluctuations, maintaining a relatively consistent standardized price throughout the period. Tether remains flat and stable, reflecting its nature as a stable coin designed to minimize volatility and maintain a consistent value.

Table 2.3: Cryptocurrency Closing Prices on 2022-12-31

Date	**Cryptocurrency**	**Closing Price**
2022-12-31	Bitcoin	16547.496094
2022-12-31	Cardano	0.246466
2022-12-31	Dogecoin	0.070294
2022-12-31	Ethereum	1196.771240
2022-12-31	Ethereum Classic	15.694440
2022-12-31	Tether	0.999674
2022-12-31	XRP	0.339929

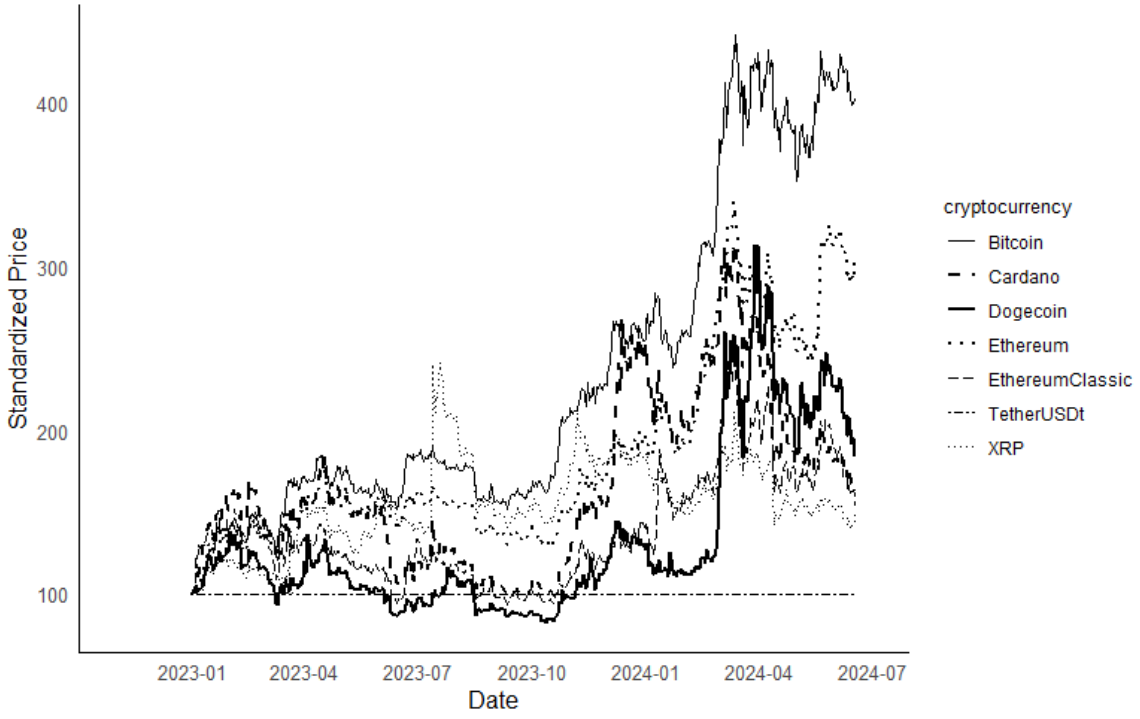


Figure 2.27: Seven-day moving average Standardized Price Since 2023

2.3 Correlation and Association

In this section, correlation matrices are computed for datasets: daily returns and standardized prices (using December 31, 2017, as the baseline). To provide a comprehensive understanding of the interrelationships among these cryptocurrencies, several correlation coefficients are calculated, including those by Pearson, Spearman, and Kendall. Additionally, the Chatterjee's nonlinear correlation is utilized, given its effectiveness in identifying oscillatory signals and its robustness to outliers and diverse distributions. This multifaceted approach aims to enhance both the reliability and depth of the correlation analysis.

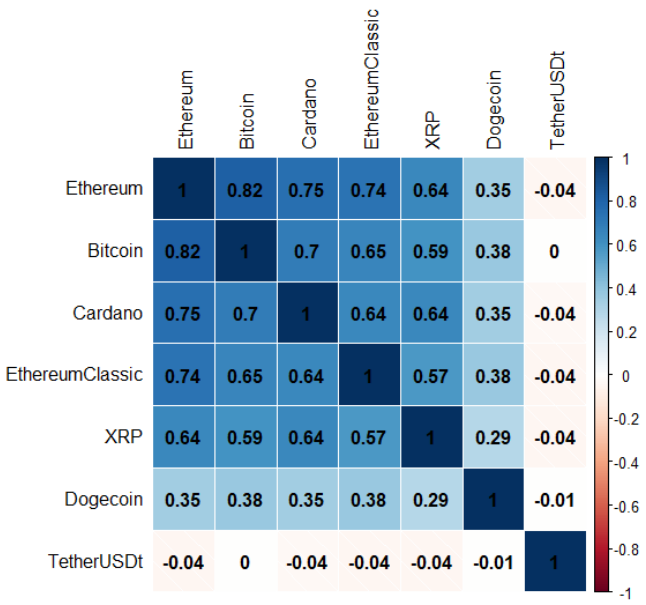


Figure 2.28: Correlation Plots for Daily Return: Pearson Correlation

The Figures 2.28 - 2.31 present correlation plots for the daily returns of the cryptocurrencies using the above four different correlation metrics. The Pearson Correlation measures the linear associations among the cryptocurrencies in Figure 2.28. High positive correlations are evident between Ethereum and Bitcoin (0.82), Ethereum and Cardano (0.75), Ethereum and Ethereum Classic (0.74), and Bitcoin and Cardano (0.70). Tether is weakly correlated with other cryptocurrencies, indicating it behaves differently in terms of daily returns. Spearman's correlation assesses monotonic relationships. The plot reflects similar patterns as in Pearson, with high correlations between Ethereum and Bitcoin (0.82), Ethereum and Ethereum Classic (0.78), Ethereum and Cardano (0.76), Ethereum and XRP (0.75), and XRP and Cardano (0.73). Tether again shows weak correlations, suggesting consistent behavior across different metrics. Kendall's tau measures any ordinal associations. The correlation strengths are generally lower compared

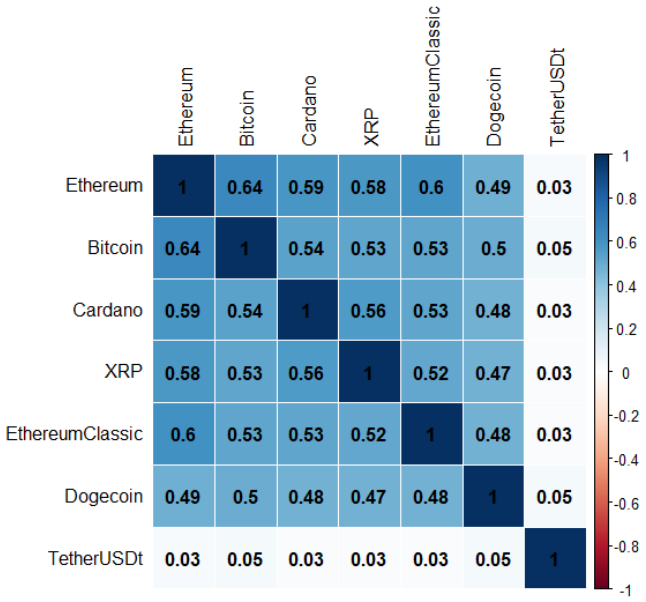


Figure 2.29: Correlation Plots for Daily Return: Kendall Correlation

to Pearson and Spearman, but the relative patterns remain similar, such as Ethereum with Bitcoin (0.64) and Ethereum with Ethereum Classic (0.60). Tether again exhibits weak correlations, reinforcing its distinctive behavior. Chatterjee's correlation is designed to detect oscillatory signals and is robust to outliers. The correlations appear lower across the board. For example, Chatterjee's correlation of Ethereum with Bitcoin is 0.49 and of Ethereum with Ethereum Classic is 0.44. It further confirms Tether's weak association with other assets. Overall, the correlation plots reveal strong associations among most cryptocurrencies, particularly Ethereum, Bitcoin, Ethereum Classic, and Cardano, while Tether consistently shows weak correlations, possibly due to its stablecoin nature.

The Figures 2.32 to 2.35 display correlation plots for the standardized prices of these cryptocurrencies. High positive Pearson correlations are observed between Ethereum and Bitcoin (0.93), Dogecoin and Ethereum Classic (0.91), and Dogecoin and

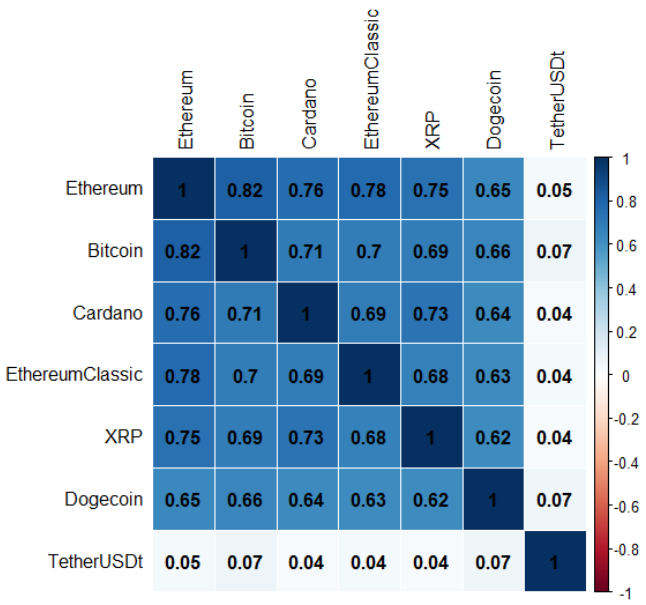


Figure 2.30: Correlation Plots for Daily Return: Spearman Correlation

Cardano (0.85). Tether shows very weak correlations with other cryptocurrencies. The Spearman's Correlation pattern is similar to Pearson, with high correlations between Ethereum and Dogecoin (0.94), Ethereum and Bitcoin (0.94), and Ethereum and Cardano (0.93). For the Kendall's correlations, the relative patterns are consistent, such as Ethereum with Dogecoin (0.79), Ethereum and Bitcoin (0.79) and Ethereum with Cardano (0.79). Higher Chatterjee's correlations are observed among Bitcoin, Ethereum, Ethereum Classic, Dogecoin and Cardano. See the upper left submatrix in Figure 2.35. Overall, the correlation plots reveal strong associations among the standardized prices of most cryptocurrencies, except with Tether.

2.4 Tests for Normality of Daily Returns

To assess whether the daily return distributions of the selected cryptocurrencies follow a normal distribution, we first apply two widely used statistical tests: the

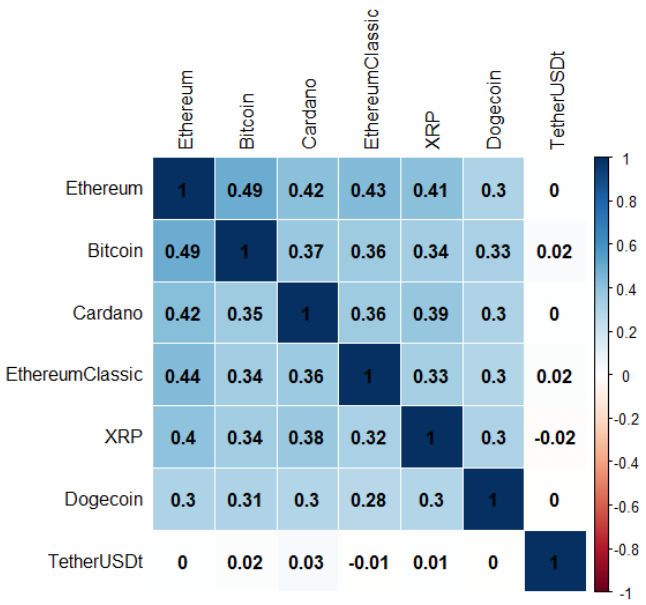


Figure 2.31: Correlation Plots for Daily Return: Chatterjee Correlation

Shapiro–Wilk test and the Kolmogorov–Smirnov (K–S) test. These tests are conducted individually to test the marginal normality for each of the seven cryptocurrencies. Shapiro and Wilk (1965) introduced the Shapiro–Wilk test, which is particularly effective for detecting departures from normality in small to moderate sample sizes. It tests the null hypothesis that a given sample $\{x_1, x_2, \ldots, x_n\}$ comes from a normally distributed population. The test statistic W is defined by

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where $x_{(i)}$ denotes the i-th order statistic (i.e., the i-th smallest value in the sample) and a_i are constants derived from the expected values of order statistics of a standard normal distribution, and are expressed as

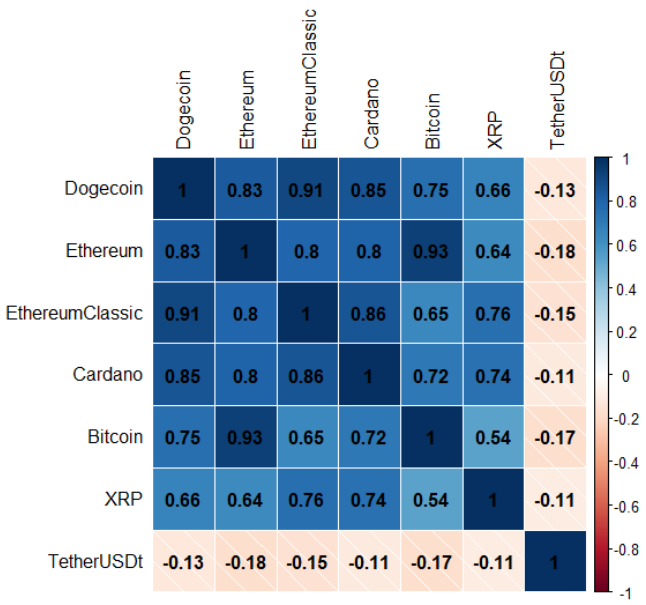


Figure 2.32: Correlation Plots for Standardized Price: Pearson Correlation

$$(a_1, \ldots, a_n) = \frac{m' V^{-1}}{(m' V^{-1} V^{-1} m)^{1/2}}$$

and $m' = (m_1, \ldots, m_n)'$ denote the vector of expected values of standard normal order statistics, and V is the corresponding covariance matrix. A small p-value resulting from the above test indicates a significant departure from normality.

The Kolmogorov–Smirnov (K–S) test is a nonparametric test used to compare the empirical cumulative distribution function $F_n(.)$ of a sample with the theoretical cumulative distribution function of a reference distribution, which is the normal distribution in this case. The test statistic D_n is defined as the maximum absolute difference between the above two,

$$D_n = \sup_x |F_n(x) - F(x)|$$

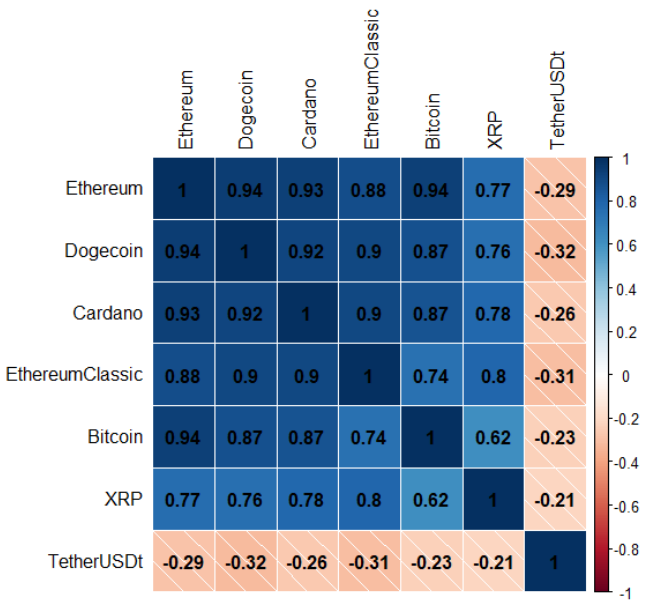


Figure 2.33: Correlation Plots for Standardized Price: Spearman Correlation

where $F_n(x)$ is given by

$$F_n(x) = \frac{\text{number of (observations} \leq x)}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{x_i \leq x\}}$$

with $\mathbf{1}_{\{x_i \leq x\}}$ is the indicator function that equals 1 if $x_i \leq x$, and 0 otherwise, and $\sup_x$ denotes the supremum over all values of x. A small p-value from the above K–S test indicates a significant deviation from the reference distribution.

Both tests are applied to the daily returns data to test the normality of the selected cryptocurrencies. The results are summarized in Table 2.4. Additionally, Q-Q plots are generated to visually evaluate the normality of these distributions, and are shown in Figures 2.36 - 2.42.

The results from both tests, along with the visual evidence from the plots, indicate that the daily returns of any of these cryptocurrencies do not follow a normal distribution.

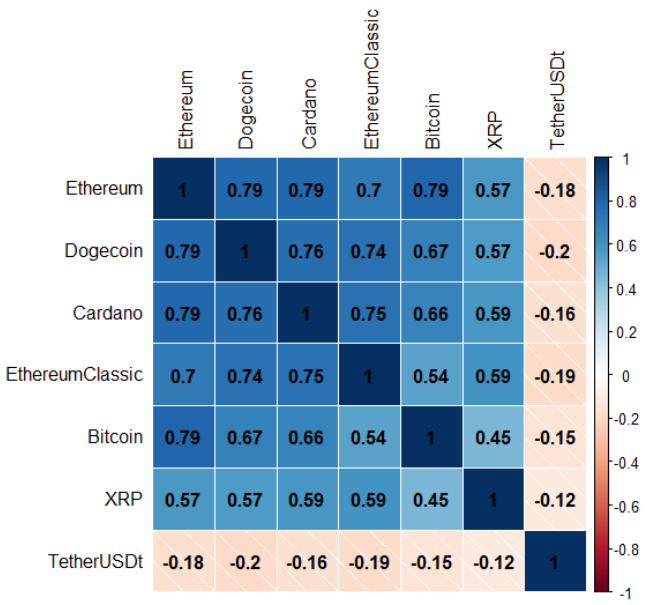


Figure 2.34: Correlation Plots for Standardized Price: Kendall Correlation

Table 2.4: P-Values for Normality Tests on Daily Returns

Cryptocurrency	**Shapiro-Wilk p-value**	**Kolmogorov-Smirnov p-value**
Bitcoin	4.204730×10^{-33}	$< 2.2 \times 10^{-16}$
Ethereum	1.108607×10^{-30}	$< 2.2 \times 10^{-16}$
Ethereum Classic	2.422156×10^{-39}	$< 2.2 \times 10^{-16}$
Cardano	7.447492×10^{-31}	$< 2.2 \times 10^{-16}$
Tether	8.661684×10^{-57}	$< 2.2 \times 10^{-16}$
XRP	4.520158×10^{-47}	$< 2.2 \times 10^{-16}$
Dogecoin	2.026260×10^{-66}	$< 2.2 \times 10^{-16}$

Subsequently, to assess the joint multivariate normality of the daily returns of six cryptocurrencies (excluding Tether), we apply Mardia's test. Originally proposed by

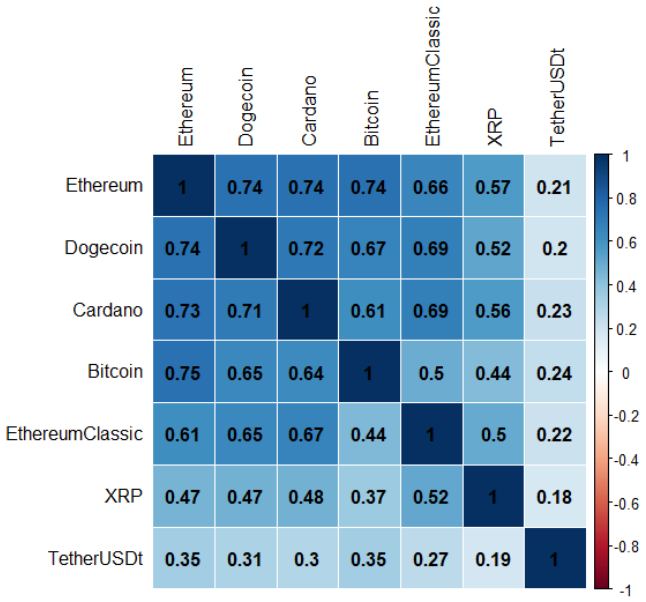


Figure 2.35: Correlation Plots for Standardized Price: Chatterjee Correlation

Mardia (1970) , this test utilizes multivariate extensions of skewness and kurtosis to evaluate whether a dataset conforms to a multivariate normal distribution.

The Mardia's multivariate skewness statistic is defined as:

$$A = \frac{1}{6n} \sum_{i=1}^{n} \sum_{j=1}^{n} \left[(x_i - \bar{x})^T S^{-1} (x_j - \bar{x}) \right]^3 \tag{2.4}$$

The Mardia's multivariate kurtosis statistic, as introduced by Mardia (1974) , is given by

$$B = \sqrt{\frac{n}{8k(k+2)}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left[(x_i - \bar{x}) S^{-1} (x_i - \bar{x}) \right]^2 - k(k+2) \right\} \tag{2.5}$$

where n is the number of observations in the dataset. x_i and x_j are the i-th and j-th observation vectors, respectively. $\bar{x}$ is the sample mean vector. S is the sample covariance

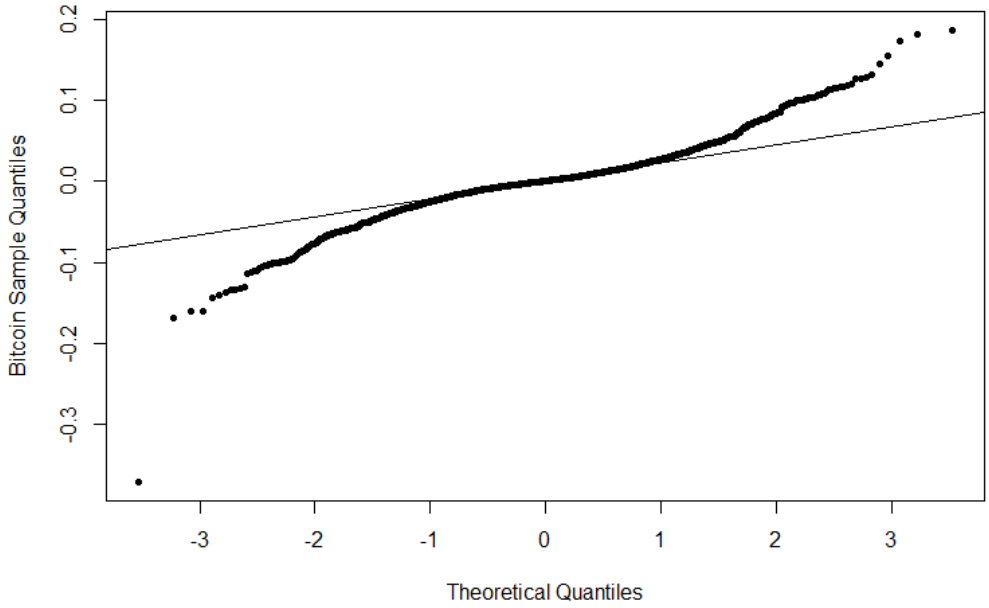


Figure 2.36: QQ Plot for Bitcoin Daily Returns

matrix, given by $S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})'$. Under the null hypothsis of multivariate normality, the skewness statistic A approximately follows a chi-square distribution with $\frac{1}{6} \cdot k(k+1)(k+2)$ degrees of freedom, and the kurtosis statistic B approximately follows a standard normal distribution $N(0,1)$.

Table 2.5: Multivariate Normality Test Results

Test	**Statistic**	**p-value**
Mardia's Skewness	29643.36	$< 2.2 \times 10^{-16}$
Mardia's Kurtosis	629.09	$< 2.2 \times 10^{-16}$

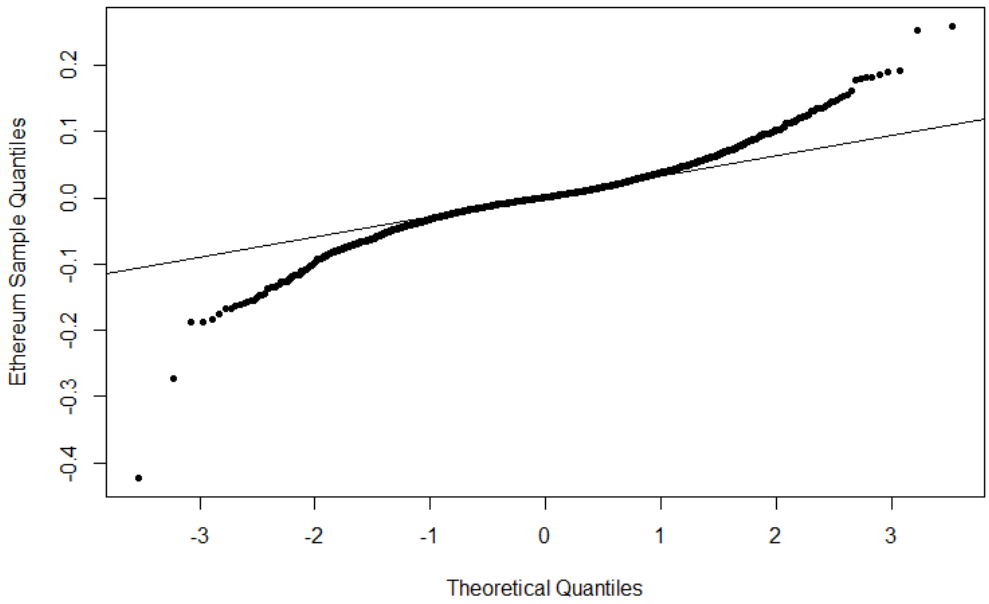


Figure 2.37: QQ Plot for Ethereum Daily Returns

To mitigate the influence of outliers, we exclude the observation corresponding to Dogecoin's maximum daily return before conducting the test. The results presented in Table 2.5 provide compelling evidence that the daily returns of the selected cryptocurrencies deviate from multivariate normality. Both the Mardia's skewness and kurtosis statistic yield p-values that are far below conventional significance thresholds (0.05). Consequently we reject the null hypothesis of multivariate normality. These results indicate significant multivariate skewness in the dataset and a substantial deviation from the expected kurtosis under the assumption of multivariate normality. This confirms that the joint distribution of the daily returns for the six cryptocurrencies (excluding Tether and the Dogecoin outlier) does not follow a multivariate normal distribution.

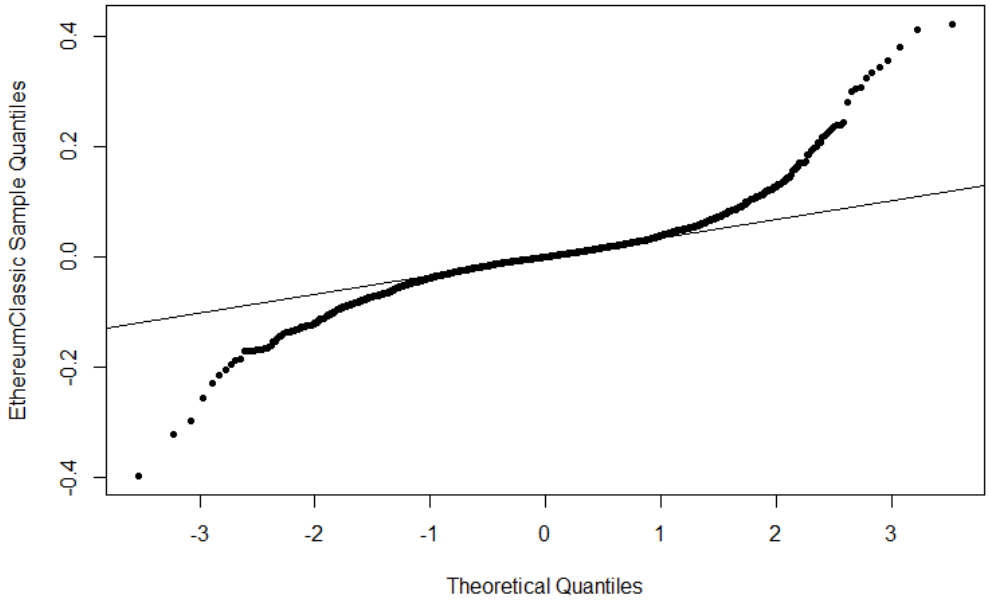


Figure 2.38: QQ Plot for Ethereum Classic Daily Returns

This conclusion is further supported by the Q-Q plot shown in Figure 2.43, which compares the squared Mahalanobis distances of the observations against the theoretical quantiles of the chi-square distribution. Under the assumption of multivariate normality, the points should closely follow the reference line. However, the observed deviations, particularly at the upper tail, indicate that the empirical distribution of the data diverges from the theoretical chi-square distribution. This visual evidence supports the conclusion drawn from Mardia's test.

We further assess multivariate normality using multivariate cumulant generating function (CGF) plots, which rely on the empirical third- and fourth-order derivatives of the CGF. For a genuinely multivariate normal distribution, all CGF derivatives of order

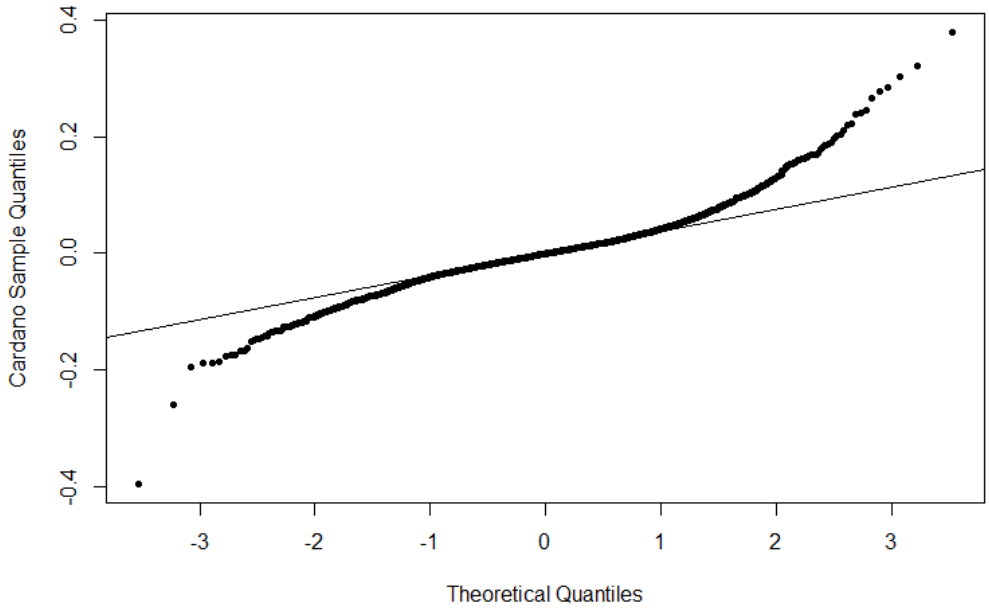


Figure 2.39: QQ Plot for Cardano Daily Returns

three and higher are identically zero. Consequently, substantial deviations or steep slopes in the empirical third- or fourth-order derivative curves signal departures from multivariate normality, see details in Ghosh (1996), Tran and Khattree (2025a, b).

Figures 2.44 and 2.45 display the CGF diagnostics based on third- and fourth-order derivatives, respectively. In both cases, the blue empirical curves show deviate substantially from the probability bands that characterize the behavior expected under true multivariate normality. These deviations provide clear evidence that the joint distribution of daily returns for the six cryptocurrencies (after excluding Tether and the Dogecoin outlier) does not conform to a multivariate normal distribution.

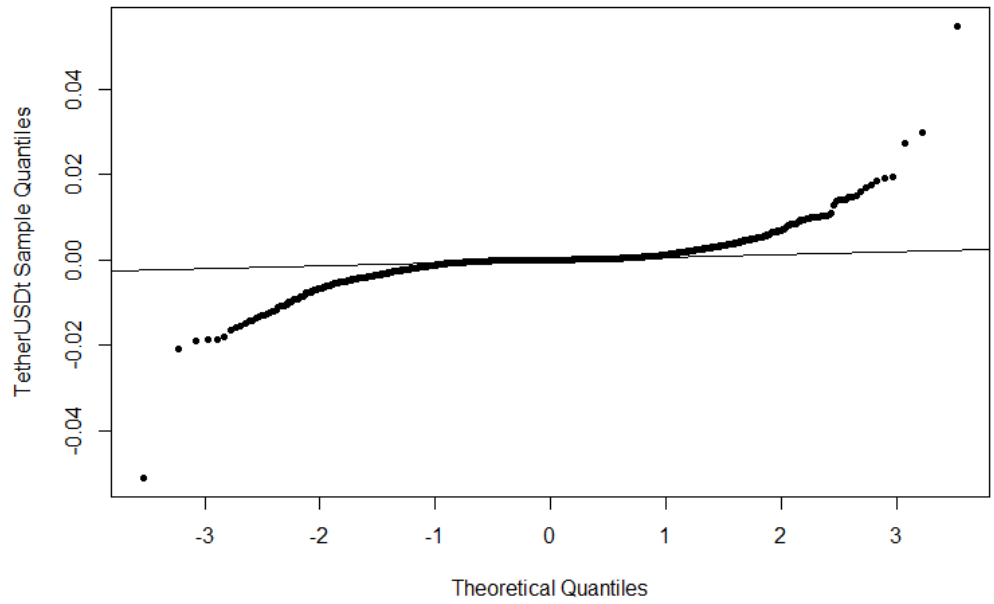


Figure 2.40: QQ Plot for Tether Daily Returns

2.5 Concluding Remarks

This chapter provides a comprehensive descriptive analysis of the statistical behavior of major cryptocurrencies over a multi-year period. Through summary statistics, visual diagnostics, and moving average trends, we identified key distributional characteristics and the presence of extreme outliers, particularly in assets like Dogecoin and XRP. Tether, in contrast, exhibits minimal variability.

The application of multiple correlation measures reveal strong interdependencies among several cryptocurrencies, especially between Bitcoin, Ethereum, and Ethereum Classic. Formal tests for normality confirm that the distributions of daily returns of all

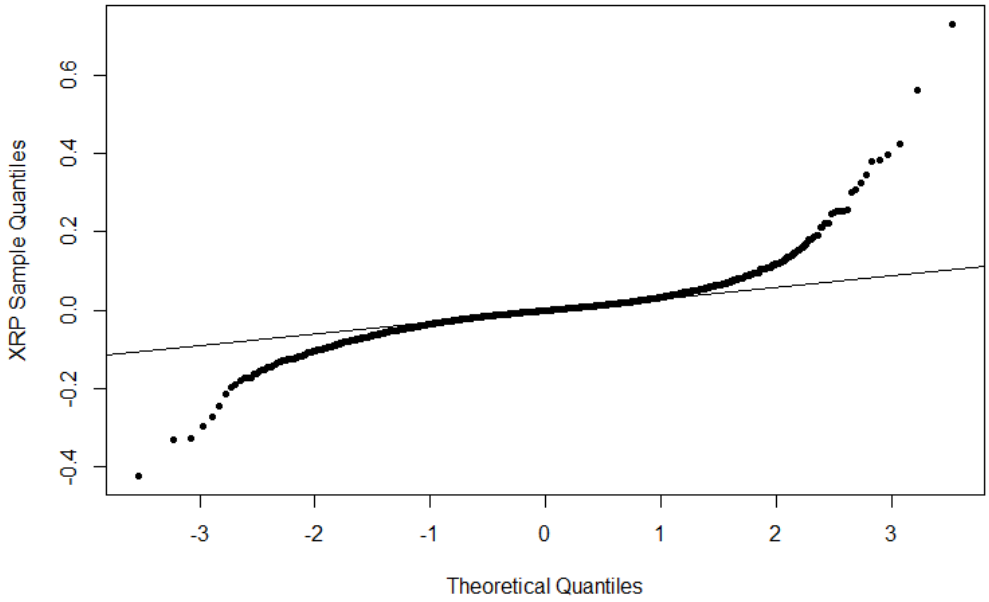


Figure 2.41: QQ Plot for XRP Daily Returns

seven cryptocurrencies deviate from normality. These findings were further supported by Q–Q plots.

Overall, the insights gained in this chapter establish an empirical foundation for the following chapters. By understanding the fundamental statistical properties and interrelationships of these digital assets, we are better equipped to model their behavior and explore strategic applications such as forecasting and pair trading.

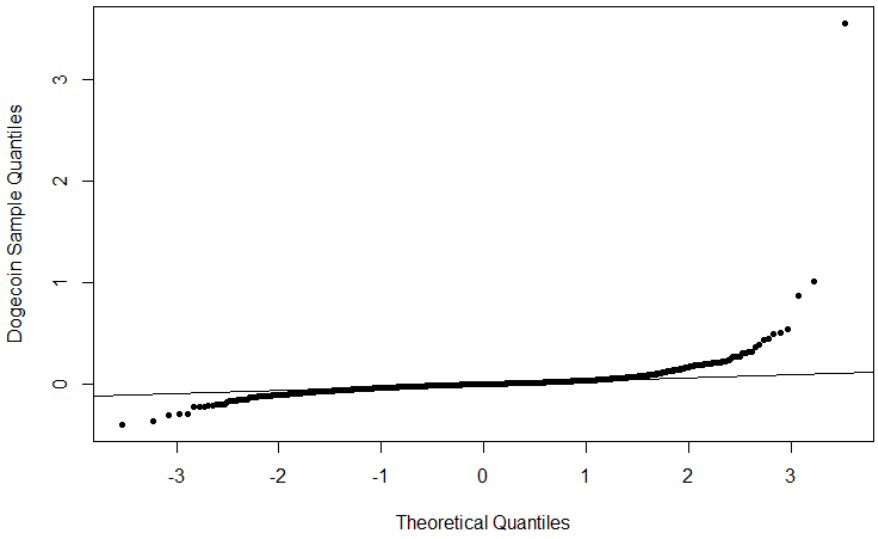


Figure 2.42: QQ Plot for Dogecoin Daily Returns

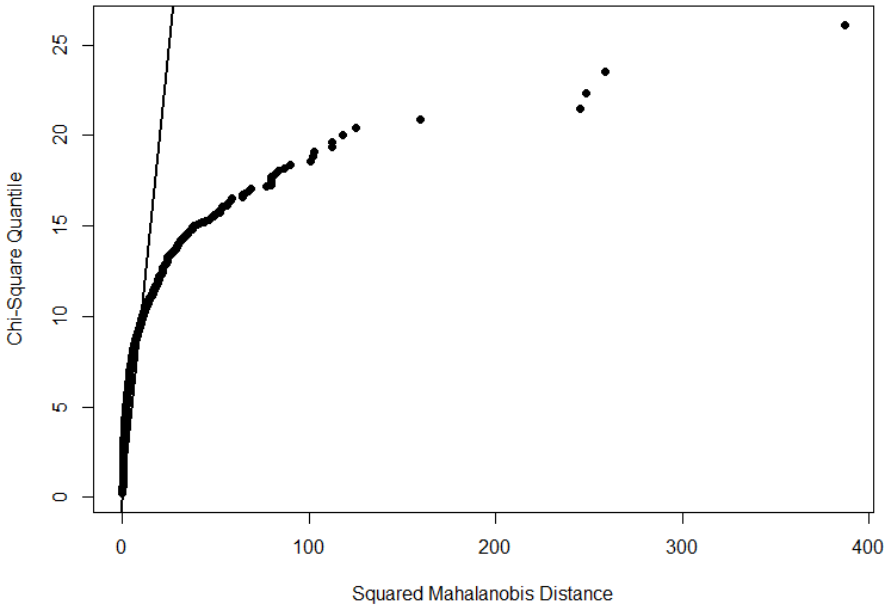


Figure 2.43: Chi-Square QQ Plot for Multivariate Normal Test

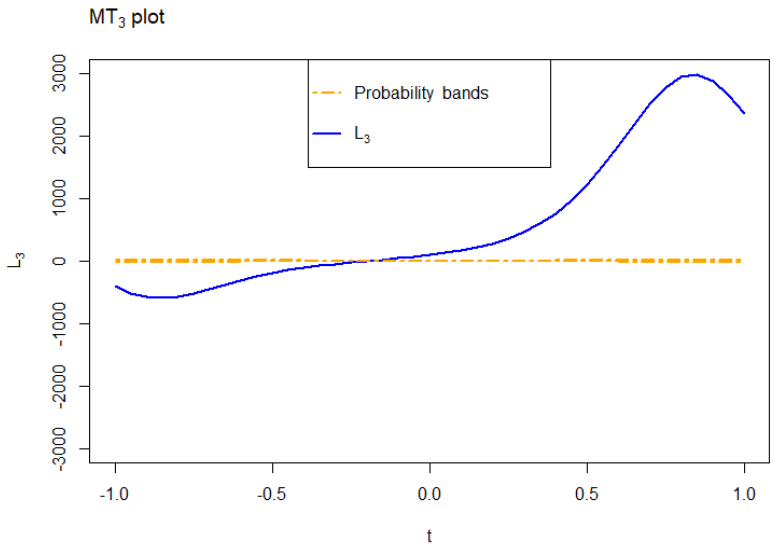


Figure 2.44: Multivariate CGF Plot Using Third-Order Derivatives

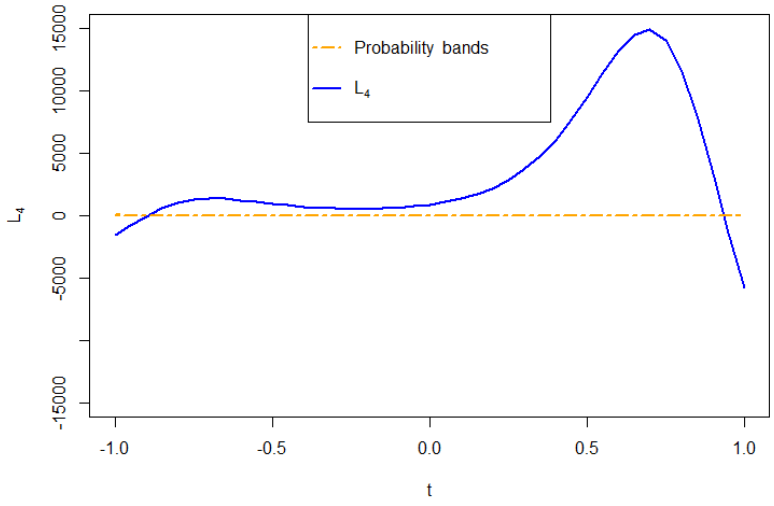


Figure 2.45: Multivariate CGF Plot Using Fourth-Order Derivatives

CHAPTER THREE

CRYPTOCURRENCIES: DESCRIPTIVE MULTIVARIATE ANALYSES

3.1 Introduction

As the cryptocurrency market continues to evolve with increasing complexity and interdependence, understanding its underlying structure requires more than univariate analyses of individual currencies. The multivariate analysis of data on several cryptocurrencies may provide more insight into their interrelationship and their collective behaviour. It may also tell us what currencies dominate the cryptomarket and if the behaviour of various cryptos is similar or quite dissimilar. The multivariate techniques thus provide a robust framework for understanding the structural relationships within the crypto market, laying the groundwork for predictive modeling and strategic applications in subsequent chapters. This chapter employs the techniques of Principal Component Analysis (PCA) and Factor Analysis to further understand the structure of the crypto data and explore the possibility of dimensionality reduction for the data. We will analyze the actual price data as well as data on daily returns.

By applying PCA, we will identify dominant components that capture much of the total variance in the data, revealing clusters of cryptocurrencies that move together and highlighting those with distinct behavior. Complementing PCA, Factor Analysis will be used to extract interpretable latent factors that may explain co-movements of cryptocurrencies.

Shah et al. (2020) propose a Principal Component Analysis based methodology to construct a dynamic cryptocurrency index that tracks the movement of the entire cryptocurrency market by adjusting the number of constituents and weights, offering a robust tool for evaluating investment returns and market volatility . Liew et al. (2019)

apply PCA to uncover the complex structures of daily return generation in the top 100 cryptocurrencies, revealing that multiple principal components are needed to explain cross-sectional variation, suggesting a multifactor structure similar to that found in hedge fund returns. Nie (2020) utilizes dimensionality reduction techniques such as Principal Component Analysis and Canonical Correlation Analysis to explore structural relationships and correlation dynamics among cryptocurrencies. Liu et al. (2019) identify the multifactor structure in cryptocurrency returns and volatility using factor analysis techniques, revealing that multiple latent factors, beyond market-wide movements, drive the cross-sectional behavior of crypto assets.

3.2 Principal Component Analysis

To uncover underlying structures and reduce the dimensionality of cryptocurrency data, we may apply Principal Component Analysis to six selected cryptocurrencies: Bitcoin, Ethereum, Ethereum Classic, XRP, Cardano, and Dogecoin. We have excluded Tether from this analysis due to its negligible variation overtime and stablecoin characteristics, which may limit its utility to any variance-based analysis.

3.2.1 Principal Component Analysis of Daily Returns

The sample correlation matrix for the daily returns of the six cryptocurrencies is presented in Table 3.1. It reveals strong positive associations among Bitcoin, Ethereum, Cardano, and Ethereum Classic. XRP has somewhat moderate correlation with the above four and a low correlation with Dogecoin. Dogecoin exhibits relatively weak correlations with all others, suggesting a very distinct behavior.

The covariance matrix for the daily return data is presented in Table 3.2, providing a quantitative measure of the variability among the cryptocurrencies.

Table 3.1: Correlation Matrix of Daily Return

	Bitcoin	Ethereum	Ethereum Classic	XRP	Cardano	Dogecoin
Bitcoin	1.00000	0.81895	0.65399	0.59355	0.69881	0.37582
Ethereum	0.81895	1.00000	0.73529	0.64199	0.75058	0.34860
Ethereum Classic	0.65399	0.73529	1.00000	0.57193	0.64334	0.37769
XRP	0.59355	0.64199	0.57193	1.00000	0.64291	0.28871
Cardano	0.69881	0.75058	0.64334	0.64291	1.00000	0.34920
Dogecoin	0.37582	0.34860	0.37769	0.28871	0.34920	1.00000

Table 3.2: Covariance Matrix of Daily Return

	Bitcoin	Ethereum	Ethereum Classic	XRP	Cardano	Dogecoin
Bitcoin	0.00128	0.00134	0.00135	0.00121	0.00136	0.00134
Ethereum	0.00134	0.00210	0.00194	0.00167	0.00187	0.00159
Ethereum Classic	0.00135	0.00194	0.00331	0.00187	0.00201	0.00216
XRP	0.00121	0.00167	0.00187	0.00322	0.00198	0.00163
Cardano	0.00136	0.00187	0.00201	0.00198	0.00295	0.00188
Dogecoin	0.00134	0.00159	0.00216	0.00163	0.00188	0.00987

3.2.1.1 Analysis of Correlation Matrix on Daily Returns As shown in Table 3.3, first three principal components account for 86.22% of the total variance upon standardization within the dataset. This substantial percent suggests that these three components can adequately represent the underlying data. See Figure 3.1.

To explore the relationship between individual principal components and individual cryptocurrencies, we examine the coefficients associated with each currency. Table 3.4 lists these for first three principal components. The first principal component by definition accounts for the highest variance(65.28%). All cryptocurrencies have positive loadings, with Ethereum (0.46085) and Bitcoin (0.44236) being the largest. Thus, as

Table 3.3: Relative Importance of Principal Components (Correlation Matrix)

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	1.97914	0.89428	0.67578	0.60470	0.54715	0.40193
Proportion of Variance	0.65283	0.13329	0.07611	0.06094	0.04990	0.02692
Cumulative Proportion	0.65283	0.78612	0.86224	0.92318	0.97308	1.00000

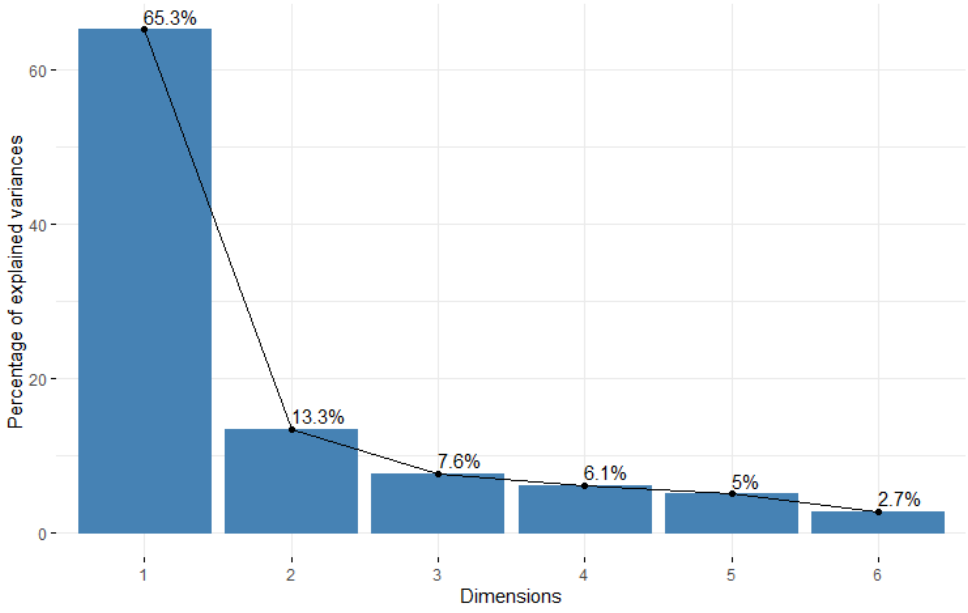


Figure 3.1: Scree Plot of Principal Components (Correlation Matrix)

weighted average the first component represents a general market trend. The second principal component, orthogonal to the first, captures about 13.33% of the variability. Dogecoin shows a large negative loading (-0.95654), whereas XRP (0.20001) and Ethereum (0.14901) have positive loadings. This component likely reflects variability specific to Dogecoin, distinguishing it from other cryptocurrencies. Regarding the third

Table 3.4: Coefficients of Currencies in First Three Principal Components (Correlation Matrix)

	PC1	**PC2**	**PC3**
Bitcoin	0.44236	0.07778	0.31616
Ethereum	0.46085	0.14901	0.25954
Ethereum Classic	0.42253	0.02123	0.31763
XRP	0.39591	0.20001	-0.84520
Cardano	0.43591	0.12777	-0.06872
Dogecoin	0.25815	-0.95654	-0.11272

principal component, XRP has a substantial negative loading (-0.84520). Loadings for Cardano and Dogecoin are smaller and those for other three are all positive. Thus this principal component distinguishes the behaviour of XRP with those of Bitcoin, Ethereum and Ethereum classic.

The Cos^2 of a variable measures how well a variable is represented by the principal components and is based on the correlation between the variable and the component (Kassambara, 2017). The Cos^2 value of variable x_j on principal component k is defined as:

$$\text{Cos}^2_{jk} = \text{Cos}^2(x_j, PC_k) = \big(\text{corr}(x_j, PC_k)\big)^2 \tag{3.1}$$

where $\text{corr}(x_j, PC_k)$ is the correlation between variable x_j and principal component PC_k and given by:

$$\text{corr}(x_j, PC_k) = e_{jk} \cdot \sqrt{\lambda_k}$$

where e_{jk} is the loading (element of the eigenvector) of variable x_j on component k, and λ_k is the eigenvalue associated with component k.

The Cos^2 metric is an indicator of the quality of representation of each cryptocurrency in the principal component space. The values closer to one indicates a stronger representation in the principal components. The sum of squared cosine values of variable x_j across the first K principal components is defined as the sum of its Cos^2 with each component:

$$\text{SCos}^2_{jK} = \text{SCos}^2(x_j, PC_K) = \sum_{k=1}^{K} \text{Cos}^2(x_j, PC_k) = \sum_{k=1}^{K} \left(\text{corr}(x_j, PC_k)\right)^2 \tag{3.2}$$

This metric quantifies how well variable x_j is represented in the subspace spanned by the first K principal components. A value close to one indicates that most of the variance of x_j is captured by these components.

The Figure 3.2 illustrates the sum of squared cosine (SCos^2) values of various cryptocurrencies with respect to the first three principal components derived from daily return data. These values reflect the quality of representation of each variable in the reduced dimensional space of the first three components. Dogecoin exhibits the highest Cos^2 value, close to 1, indicating that its variance is almost entirely captured by the first three components. XRP also shows strong representation, though slightly lower than Dogecoin. Ethereum follows with a moderately high value, suggesting substantial but not complete representation. Bitcoin, Cardano, and Ethereum Classic show lower Cos^2 values among the group, implying that their variance is less well captured in this reduced space, but still within a reasonable range.

The Figure 3.3 presents the SCos^2 values of various cryptocurrencies relative to the first two principal components. Dogecoin shows the highest Cos^2 value approaching

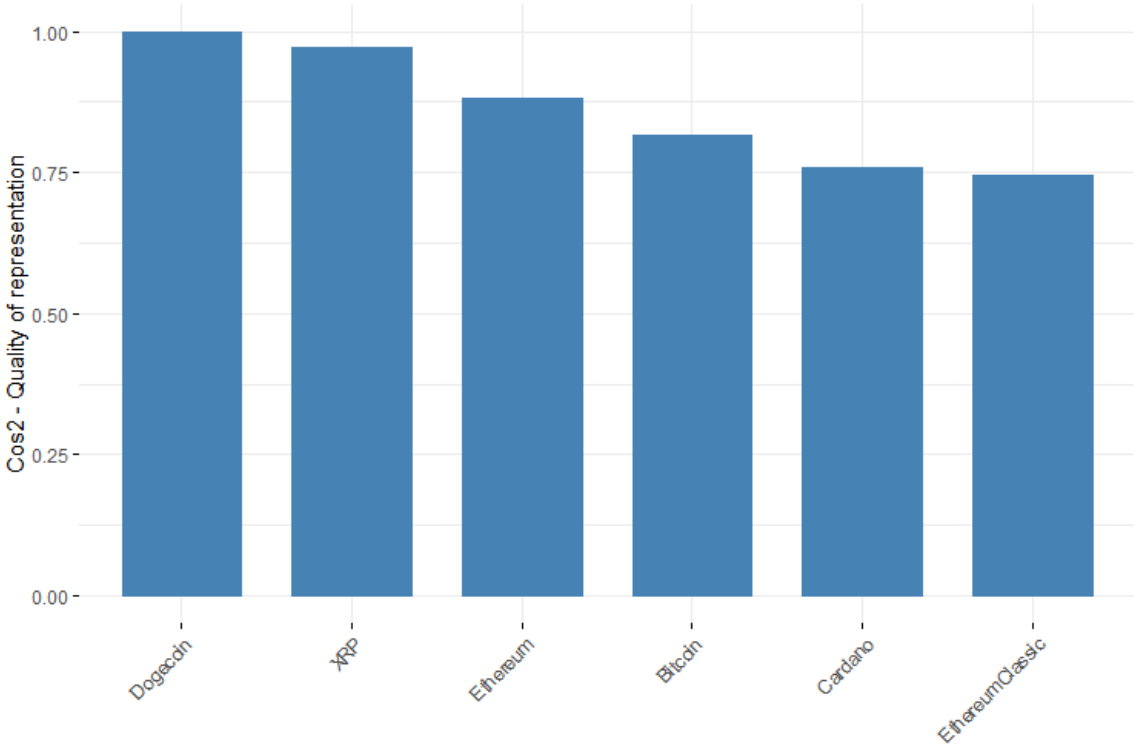


Figure 3.2: SCos^2 of Variables to Component 1 to 3 for Daily Return Data (Correlation Matrix)

1, indicating that its variance is very well captured by the first two components. Ethereum follows with a strong representation with a slightly lower Cos^2 value, suggesting that a substantial portion of its variance is explained. Bitcoin and Cardano show similar Cos^2 values, indicating moderate representation. Ethereum Classic and XRP exhibit lower Cos^2 values, implying that they are less well-represented by the first two components, though still reasonably captured.

In both plots, Dogecoin consistently shows the highest Cos^2 value, indicating that its variance is very well captured by the leading components. XRP, which had the lowest representation in the two-component plot, shows a noticeable increase in Cos^2 when the

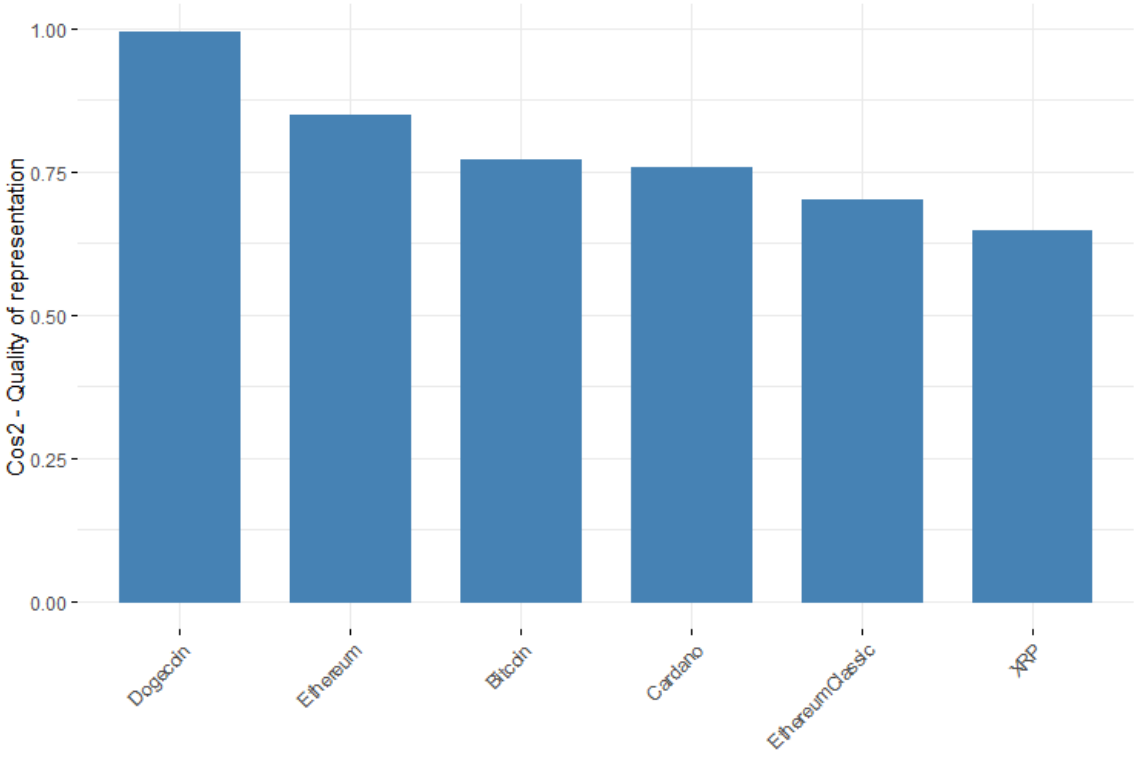


Figure 3.3: SCos^2 of Variables to Component 1 and 2 for Daily Return Data (Correlation Matrix)

third component is included, suggesting that its variance is better captured in a three-dimensional PCA space. Similarly, Ethereum Classic and Cardano also benefit from the additional component, showing improved representation.

The Figure 3.4 shows the correlation biplot for the first two principal components, emphasizing the relationship among cryptocurrencies. The arrows in the figure present cryptocurrencies, while the length and direction of arrows indicate the correlation and contribution of cryptocurrencies with the principal components. The PC1 (Dim1) captures 65.3% of the variance in the dataset. Cryptocurrencies like Ethereum, XRP, Cardano, Ethereum Classic, and Bitcoin have vectors closely aligned with PC1, indicating that most

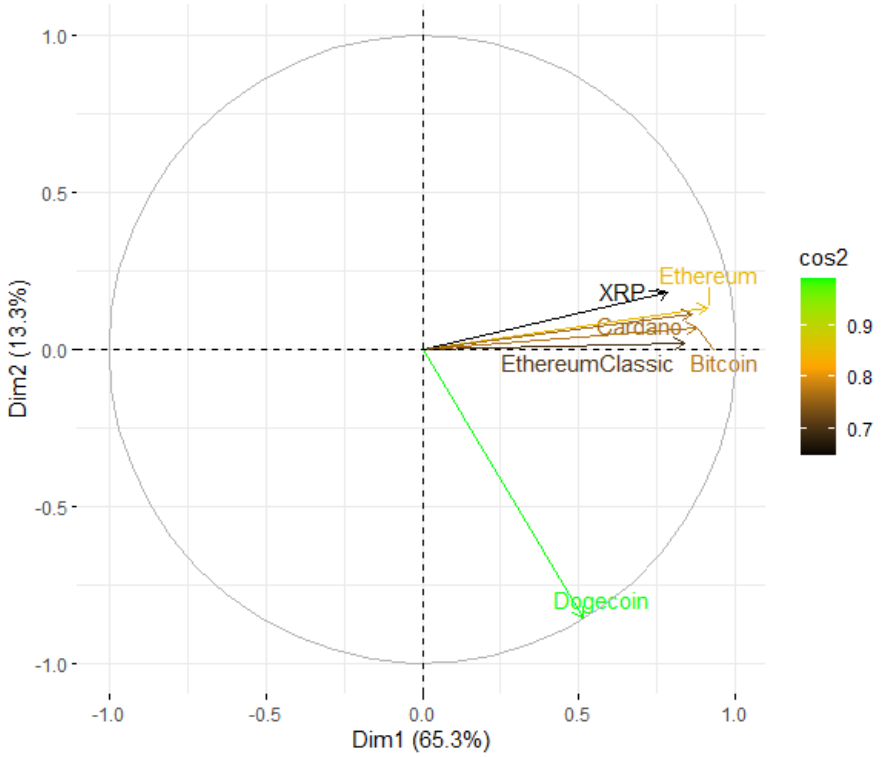


Figure 3.4: Biplot for Daily Return Data (Correlation Matrix)

of their variability is explained by this component. The arrow for Dogecoin extends prominently along PC2 and is colored green, indicating a high Cos^2 value. This suggests that Dogecoin has a unique pattern of variability compared to other cryptocurrencies, contributing significantly to PC2. Other Cryptocurrencies Ethereum, XRP, Cardano, Ethereum Classic, and Bitcoin have shorter arrows with a more clustered orientation towards PC1. Their similar direction implies a shared underlying trend affecting these currencies, primarily captured by PC1.

3.2.1.2 Analysis of Covariance Matrix on Daily Returns The results of the principal component analysis conducted using the covariance matrix, as presented in Table 3.5,

indicate that the first principal component captures approximately 59.6% of the total variance, suggesting a dominant latent structure within the dataset. The second component contributes an additional 25.5%, resulting in a cumulative explained variance of 85.0% with just two components. This substantial coverage, visualized in Figure 3.5, suggests that the original six-dimensional data can be effectively reduced to two dimensions with minimal loss of information. The remaining components account for smaller proportions of variance and may be considered less critical for data representation. As the PCA was performed on the covariance matrix, the influence of variable scale is preserved, allowing variables with larger variances to exert greater influence on the principal components.

Table 3.5: Relative Importance of Principal Components(Covariance Matrix)

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
Standard deviation	0.11637	0.07605	0.03763	0.03245	0.02550	0.01666
Proportion of Variance	0.59597	0.25452	0.06233	0.04634	0.02862	0.01221
Cumulative Proportion	0.59597	0.85050	0.91283	0.95917	0.98779	1.00000

The coefficients presented in Table 3.6 represent the contributions of each cryptocurrency to the first three principal components. In the first component, all cryptocurrencies exhibit positive loadings, with Dogecoin showing the highest value (0.72610), suggesting that this component captures a general market-wide movement in which Dogecoin plays an influential role. The second component reveals a contrasting structure, characterized by a negative loading for Dogecoin (-0.68575) and moderate positive loadings for the remaining assets, indicating that this component may represent a

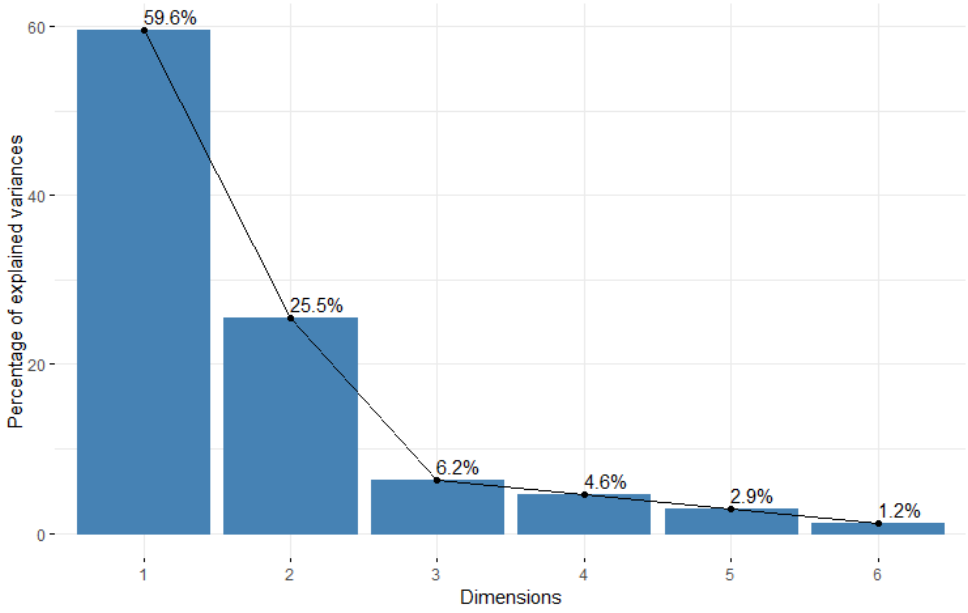


Figure 3.5: Scree Plot of Principal Components (Covariance Matrix)

Table 3.6: Coefficients of Currencies in Principal Components (Covariance Matrix)

	PC1	**PC2**	**PC3**
Bitcoin	0.21827	0.20197	0.09067
Ethereum	0.28820	0.31262	0.16688
Ethereum Classic	0.36015	0.33861	0.60852
XRP	0.31605	0.38599	-0.76843
Cardano	0.33539	0.35719	-0.03213
Dogecoin	0.72610	-0.68575	-0.04601

divergence between Dogecoin and the other five cryptocurrencies. The third component is dominated by a substantial negative loading for XRP (-0.76843), alongside moderate to large positive loadings for Ethereum and Ethereum Classic, suggesting this component

may capture a latent factor differentiating XRP from these two assets. The comparison between the principal components derived from the correlation and covariance matrices reveals structural similarities and key differences. While the overall direction of the components is largely preserved, the loadings show notable shifts, especially for assets with relatively higher variance like Dogecoin. In the covariance-based PCA, Dogecoin dominates the first component due to its large variance, significantly altering the component structure compared to the correlation-based PCA.

In the context of principal component analysis based on the correlation matrix, Cos^2 values represent the squared correlations between variables and principal components. For each variable, the sum of its Cos^2 values across all components equals 1, enabling a meaningful interpretation of how well the variable is represented in the reduced dimensional space. However, when PCA is performed using the covariance matrix, the original scales and variances of the variables are preserved. In this case, the sum of a variable's Cos^2 values across all principal components equals the variable's original variance. A high Cos^2 value indicates that the corresponding principal component explains a large portion of the variable's original variance.

We demonstrate this property in the following proof. Let X be a centered data matrix of size $n \times p$:

$$X = (x_1, x_2, \ldots, x_p)$$

where n is the number of observations and p is the number of variables. Each column of X has mean zero. The sample covariance matrix is:

$$C = X'X$$

and the variance of variable x_j is:

$$C_{jj} = \text{Var}(x_j)$$

Principal Component Analysis is performed via the eigen-decomposition of C:

$$C = E\Lambda E'$$

where $E = [e_1, e_2, \ldots, e_p]$ is an orthogonal matrix whose columns are the eigenvectors, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ is a diagonal matrix of eigenvalues. From the property of the eigen-decomposition of a symmetric matrix, we can rewrite:

$$C = E\Lambda E' = \begin{bmatrix} e_1 & e_2 & \cdots & e_p \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} e'_1 \\ e'_2 \\ \vdots \\ e'_p \end{bmatrix} = \sum_{k=1}^{p} \lambda_k e_k e'_k$$

For variable x_j, the variance is given by the j-th diagonal element of C:

$$\text{Var}(x_j) = C_{jj} = \sum_{k=1}^{p} \lambda_k (e_k e'_k)_{jj} = \sum_{k=1}^{p} \lambda_k e_{jk}^2$$

This follows from the matrix product:

$$e_k e'_k = \begin{bmatrix} e_{1k} \\ e_{2k} \\ \vdots \\ e_{pk} \end{bmatrix} \begin{bmatrix} e_{1k} & e_{2k} & \cdots & e_{pk} \end{bmatrix} = \begin{bmatrix} e_{1k}^2 & e_{1k}e_{2k} & \cdots & e_{1k}e_{pk} \\ e_{2k}e_{1k} & e_{2k}^2 & \cdots & e_{2k}e_{pk} \\ \vdots & \vdots & \ddots & \vdots \\ e_{pk}e_{1k} & e_{pk}e_{2k} & \cdots & e_{pk}^2 \end{bmatrix}$$

The diagonal element $(e_k e'_k)_{jj}$ is simply e_{jk}^2, which confirms:

$$\text{Var}(x_j) = \sum_{k=1}^{p} \lambda_k e_{jk}^2$$

The squared cosine (Cos^2) of variable x_j on principal component k is given by

$$\text{Cos}^2_{jk} = \left(\text{corr}(x_j, \text{PC}_k)\right)^2 = \left(e_{jk} \cdot \sqrt{\lambda_k}\right)^2 = e^2_{jk}\lambda_k$$

Then, the sum of squared cosine values for variable x_j across all principal components equals its original variance:

$$\sum_{k=1}^{p} \text{Cos}^2_{jk} = \sum_{k=1}^{p} e^2_{jk}\lambda_k = \text{Var}(x_j)$$

Thus, Cos^2 values quantify how much of the variable's variance is captured by each principal component. In the case of PCA based on the correlation matrix, the variables are standardized such that $\text{Var}(x_j) = 1$, and therefore:

$$\sum_{k=1}^{p} \cos^2_{jk} = 1$$

when PCA is based on the covariance matrix, the original variances are preserved. To interpret Cos^2 values as proportions of explained variance and to compare how well different variables are represented by a specific principal component, we can standardize the Cos^2 values by dividing them by the variable's variance:

$$\text{Standardized } \text{Cos}^2_{jk} = \frac{\text{Cos}^2_{jk}}{\text{Var}(x_j)} = \text{Proportion of variance of } x_j \text{ explained by } \text{PC}_k$$

Accordingly, to quantify how well a variable x_j is represented by the first K principal components, we can standardize the sum of squared cosine values across the first K components:

$$\text{Standardized } \text{SCos}^2_{jK} = \frac{\text{SCos}^2_{jK}}{\text{Var}(x_j)} = \text{Proportion of variance of } x_j \text{ explained by the first } K \text{ PCs} \tag{3.3}$$

The Figure 3.6 presents the standardized squared cosine (SCos^2) values of cryptocurrencies with respect to the first two principal components derived from daily

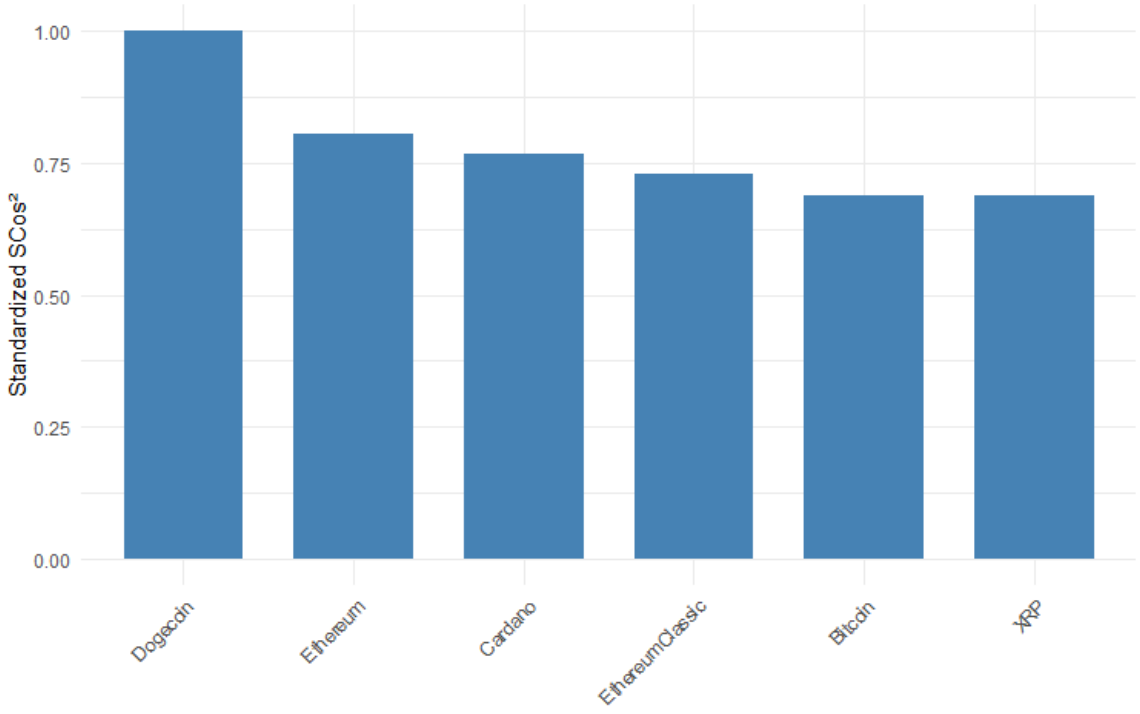


Figure 3.6: Standardized $SCos^2$ of Variables to Component 1 and 2 for Daily Return Data (Covariance Matrix)

return data using the covariance matrix. Dogecoin exhibits the highest $SCos^2$ value close to 1, indicating that its variance is mostly captured by the first two components. Ethereum follows with a slightly lower $SCos^2$ value, suggesting that a substantial portion of its variance is also explained. Cardano and EthereumClassic show moderate representation, while Bitcoin and XRP have the lowest $SCos^2$ values among the group, implying that their return behaviors are less well-represented in the first two components, though still within an acceptable range.

The Figure 3.7 extends the analysis to the first three principal components. Dogecoin continues to exhibit the highest $SCos^2$ value, reaffirming its strong alignment

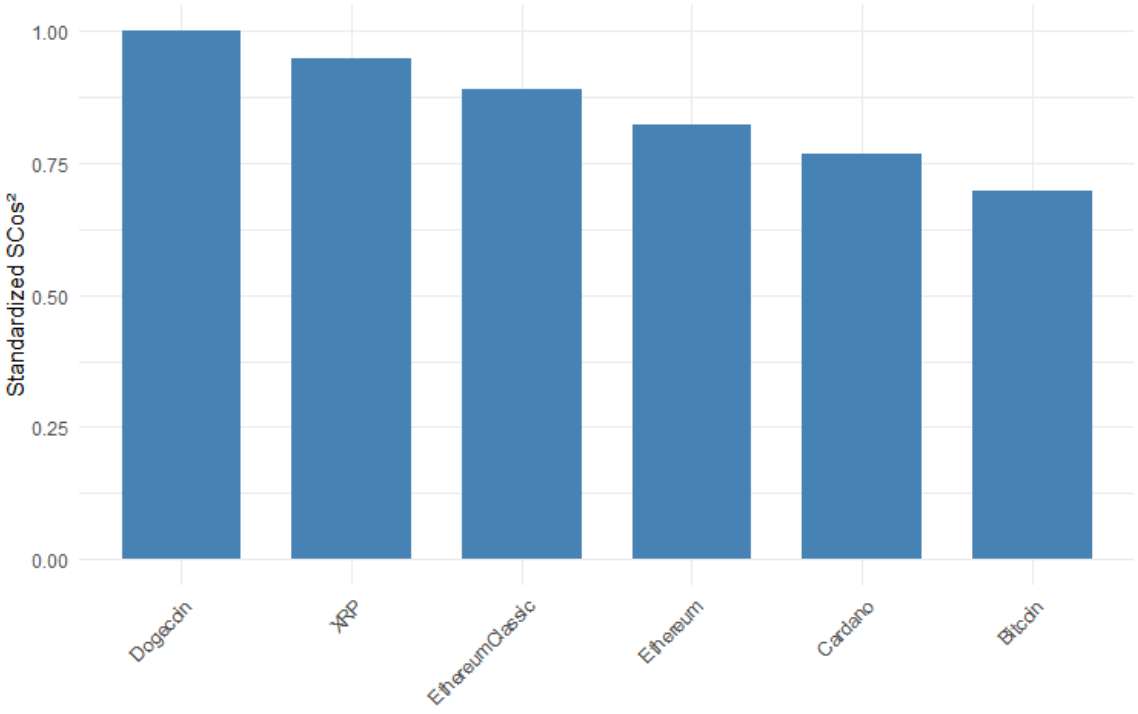


Figure 3.7: Standardized $SCos^2$ of Variables to Component 1 to 3 for Daily Return Data (Covariance Matrix)

with the dominant variance structure. Notably, XRP, which had the lowest representation in the two-component plot, shows a marked increase in $SCos^2$ when the third component is included, indicating that its variance is better captured in a three-dimensional PCA space. Similarly, EthereumClassic and Cardano also benefit from the inclusion of the third component, showing improved representation.

3.2.2 Principal Component Analysis of Standardized Prices

To further explore the structural relationships among cryptocurrencies, we also perform the Principal Component Analysis on the standardized price data. Standardization

is performed by scaling each cryptocurrency's price relative to its value on the initial observation date 12/31/2017(price on 12/31/2017 is taken as 100), allowing for meaningful comparisons across assets with vastly different price scales. The correlation and covariance matrix of the standardized price data are presented in Tables 3.7 and 3.8. The correlation matrix shows strong positive associations among most cryptocurrencies, particularly between Ethereum and Bitcoin, and Ethereum Classic and Dogecoin. In the covariance matrix, we observed especially high variances for Dogecoin and Bitcoin.

Table 3.7: Correlation Matrix of Standardized Prices

	Bitcoin	Ethereum	Ethereum Classic	XRP	Cardano	Dogecoin
Bitcoin	1.00000	0.93023	0.64474	0.48202	0.71674	0.75306
Ethereum	0.93023	1.00000	0.79357	0.57772	0.79879	0.83488
Ethereum Classic	0.64474	0.79357	1.00000	0.71638	0.85748	0.90397
XRP	0.48202	0.57772	0.71638	1.00000	0.69949	0.59640
Cardano	0.71674	0.79879	0.85748	0.69949	1.00000	0.84777
Dogecoin	0.75306	0.83488	0.90397	0.59640	0.84777	1.00000

Table 3.8: Covariance Matrix of Standardized Prices

	Bitcoin	Ethereum	Ethereum Classic	XRP	Cardano	Dogecoin
Bitcoin	16965.09	19131.10	4880.23	872.39	7250.49	98348.37
Ethereum	19131.10	24931.34	7281.79	1267.53	9795.69	132175.55
Ethereum Classic	4880.23	7281.79	3377.20	578.48	3870.15	52673.20
XRP	872.39	1267.53	578.48	193.08	754.88	8309.20
Cardano	7250.49	9795.69	3870.15	754.88	6031.91	66018.02
Dogecoin	98348.37	132175.55	52673.20	8309.20	66018.02	1005341.60

Table 3.9: Relative Importance of Principal Components for Standardized Prices (Correlation Matrix)

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	2.17759	0.79962	0.58504	0.39629	0.28755	0.19164
Proportion of Variance	0.79031	0.10656	0.05705	0.02617	0.01378	0.00612
Cumulative Proportion	0.79031	0.89688	0.95392	0.98010	0.99388	1.00000

3.2.2.1 Analysis of Correlation Matrix on Standardized Prices The Principal Component Analysis of correlation matrix results in Table 3.9 provide some insights into the importance of different components for the standardized price data. PC1 has the highest standard deviation (2.17759), suggesting it captures the most of the total variance on standardized scale in the dataset. Accordingly, it explains 79.03% of the total variance. As seen in Table 3.10, all coefficients are positive, thus as a weighted average, it represents an overall market trend of all six cryptocurrencies. The second component captures an additional 10.66% of the variance and reveals contrasting behavior between two sets of currencies, Bitcoin (0.55129) and Ethereum (0.36263) load positively, while XRP (-0.69928) and Ethereum Classic (-0.23301) load negatively. This component may reflect a divergent price dynamics between speculative (XRP and Ethereum Classic) and more stable (Bitcoin and Ethereum) assets. The third captures 5.70% of the total variance and is dominated by XRP (0.58029), Bitcoin (0.43066), Dogecoin (-0.47386) and Ethereum Classic (-0.42245). This component likely captures the difference in the behavior of first two and last two over time. Together, the first three components explain 95.39% of the total variance, indicating that three components are sufficient to capture most of the information in the data.

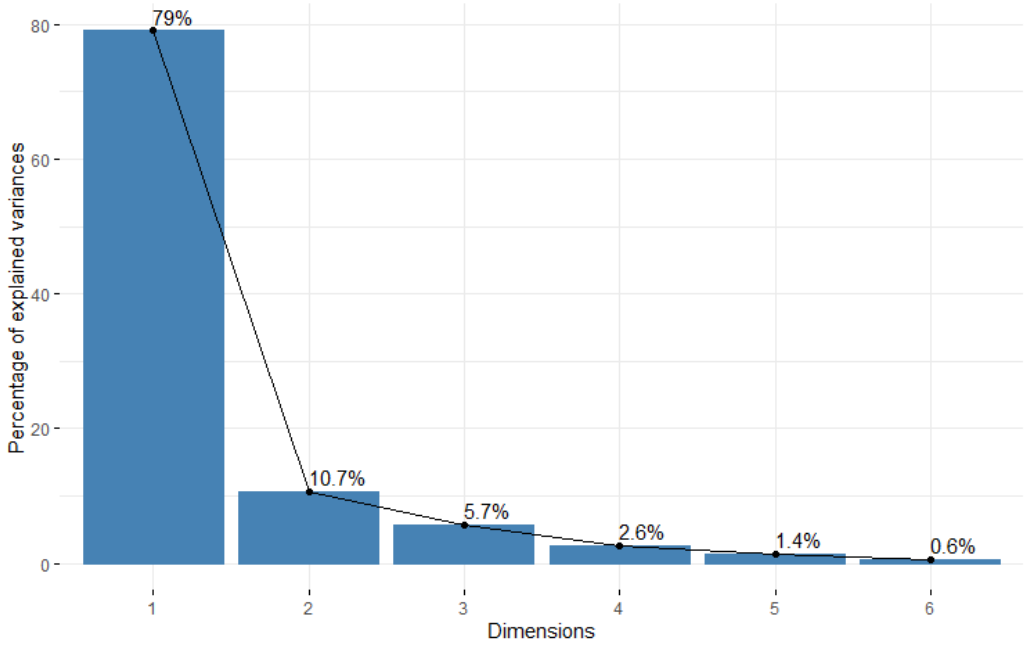


Figure 3.8: Scree Plot of Principal Components for Standardized Prices (Correlation Matrix)

Table 3.10: Coefficients of Currencies in First Three Principal Components (Correlation Matrix)

	PC1	**PC2**	**PC3**
Bitcoin	0.39156	0.55129	0.43066
Ethereum	0.42715	0.36263	0.20801
Ethereum Classic	0.42509	-0.23301	-0.42245
XRP	0.34551	-0.69928	0.58029
Cardano	0.42520	-0.13247	-0.17753
Dogecoin	0.42820	0.06124	-0.47386

The Figure 3.9 presents the sum of Cos^2 values of cryptocurrencies with respect to the first three principal components derived from standardized price data. The majority of

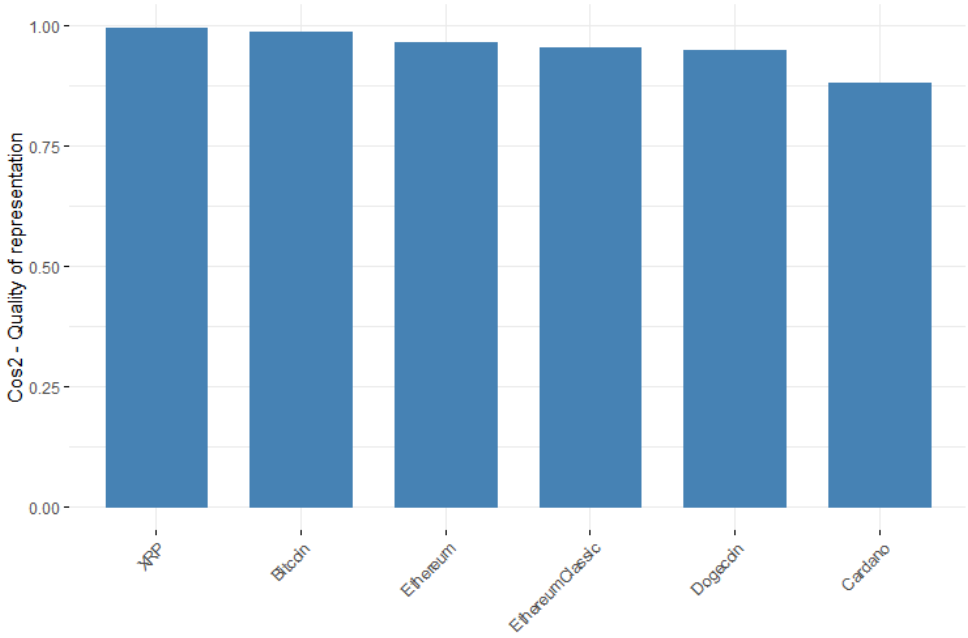


Figure 3.9: SCos^2 of Variables to Component 1 to 3 for Standardized Price Data (Correlation Matrix)

variables exhibit high Cos^2 values approaching one, indicating that the first three components collectively capture most of the variance associated with each cryptocurrency. While Dogecoin and Cardano show slightly lower Cos^2 values compared to the other assets, their representation remains robust

The Figure 3.10 displays the Cos^2 plot the cryptocurrencies relative to the first two principal components. Ethereum and Bitcoin have the highest Cos^2 values, indicating that they are very well represented by the first two principal components. Other four cryptocurrencies have slightly lower Cos^2 values, but still well captured. Compared to the plot for three-components, the Cos^2 values are slightly lower across the board, particularly for XRP and Bitcoin, suggesting that a portion of their variance lies outside the first two components.

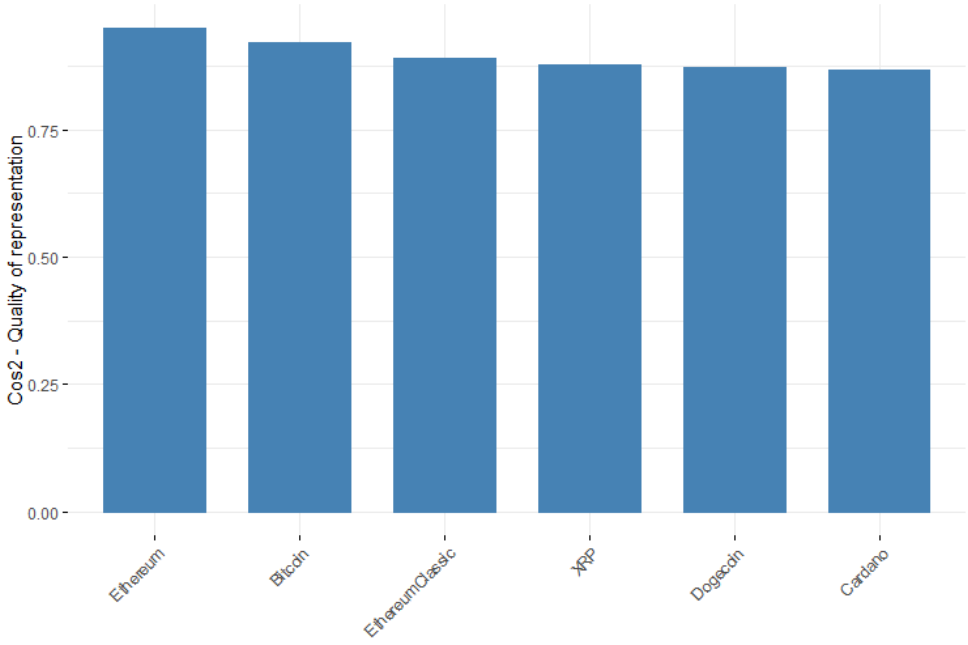


Figure 3.10: $SCos^2$ of Variables to Component 1 and 2 for Standardized Price Data (Correlation Matrix)

The Figure 3.11 presents a biplot for cryptocurrencies derived from the standardized price data. The plot illustrates the distribution of variable vectors across the first two principal components. The vectors representing Bitcoin and Ethereum are oriented closely together and show a high Cos^2 values, suggesting that these cryptocurrencies exhibit similar price movements and are strongly correlated with PC1. Dogecoin and Cardano also have similar directional trends but are less strongly represented. Ethereum Classic and XRP, with arrows pointing in differing directions, suggest distinct price movement characteristics from the other cryptocurrencies.

3.2.2.2 Analysis of Covariance Matrix on Standardized Prices Using the covariance matrix, the PCA result for the standardized prices is presented in Table 3.11, indicating

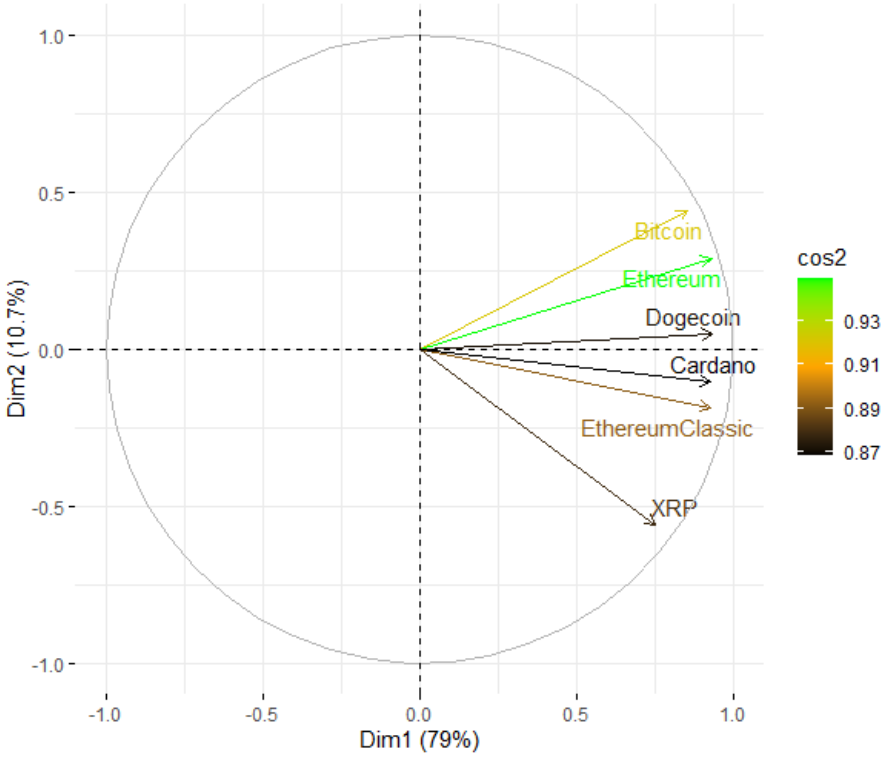


Figure 3.11: Biplot for Standardize Price Data (Correlation Matrix)

Table 3.11: Importance of Components for Standardized Price (Covariance Matrix)

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
Standard deviation	1019.54	115.81	43.998	34.202	18.369	8.803
Proportion of Variance	0.98397	0.01270	0.00183	0.00111	0.00032	0.00007
Cumulative Proportion	0.98397	0.99667	0.99850	0.99961	0.99993	1.00000

that the first principal component overwhelmingly dominates the variance structure, accounting for approximately 98.4% of the total variance. The second component contributes only 1.27%, and the remaining components each explain less than 0.2% of the

variance. See Figure 3.12. The cumulative proportion reaches 99.67% with just two components, implying that dimensionality reduction to two components would retain nearly all information. Since the PCA was performed on the covariance matrix, the results are heavily influenced by the absolute magnitudes of the variables, and the first component may be dominated by the asset with the largest variance.

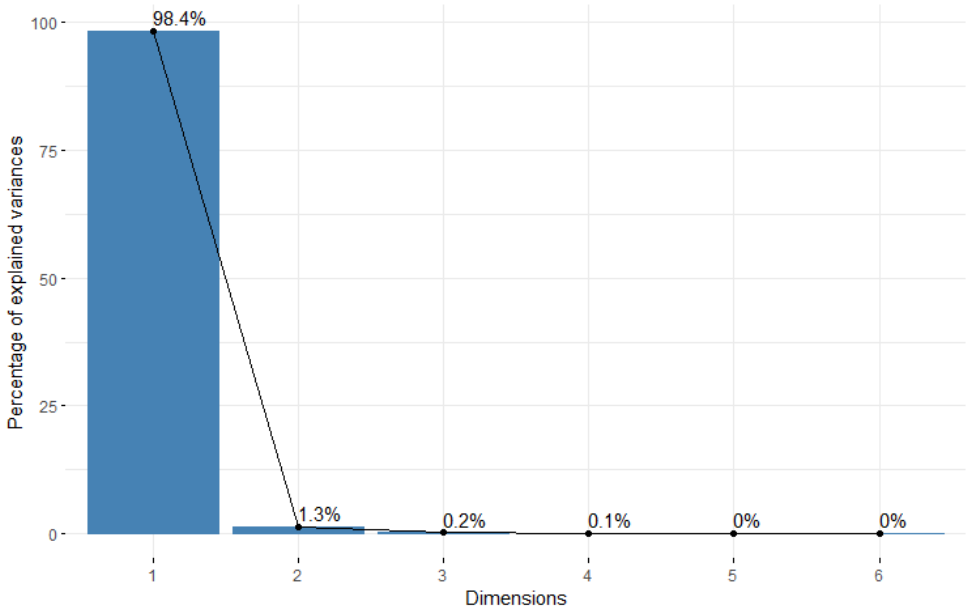


Figure 3.12: Scree Plot of Principal Components for Standardized Prices (Covariance Matrix)

As shown in Table 3.12, the first component is overwhelmingly dominated by Dogecoin, with a loading of 0.98305, indicating that this component primarily captures the variance associated with Dogecoin's price movements. The other assets contribute minimally to this component. The second component is characterized by strong positive

Table 3.12: Coefficients of Currencies in Various Principal Components (Covariance Matrix)

	PC1	PC2	PC3
Bitcoin	0.09767	0.68912	0.42274
Ethereum	0.13087	0.69758	-0.29271
Ethereum Classic	0.05158	-0.00109	-0.40638
XRP	0.00817	0.01244	-0.13140
Cardano	0.06490	0.10055	-0.74064
Dogecoin	0.98305	-0.16802	0.06827

loadings for Bitcoin (0.68912) and Ethereum (0.69758), indicating that this component captures a shared pattern between these two major cryptocurrencies. Dogecoin, in contrast, has a negative loading (-0.16802), suggesting a divergence from Bitcoin and Ethereum in this dimension. The third component shows a strong negative loading for Cardano (-0.74064), and moderate contributions from Bitcoin (0.42274) and Ethereum Classic (-0.40638), suggesting that this component may reflect asset-specific volatility patterns. These loadings are heavily influenced by the absolute magnitudes of the variables, with Dogecoin dominating the first component due to its scale. In comparison with the results, the correlation-based PCA provides a more balanced view of asset relationships, with the first principal component explaining 79% of the variance and all assets contributing relatively evenly, reflecting underlying co-movement patterns. In contrast, the covariance-based PCA is heavily dominated by Dogecoin, which alone accounts for over 98% of the variance in the first component due to its large price scale, leading to a component structure that captures magnitude rather than relational dynamics.

The Figure 3.13 presents the standardized $SCos^2$ values of cryptocurrencies with respect to the first two principal components derived from standardized price data using

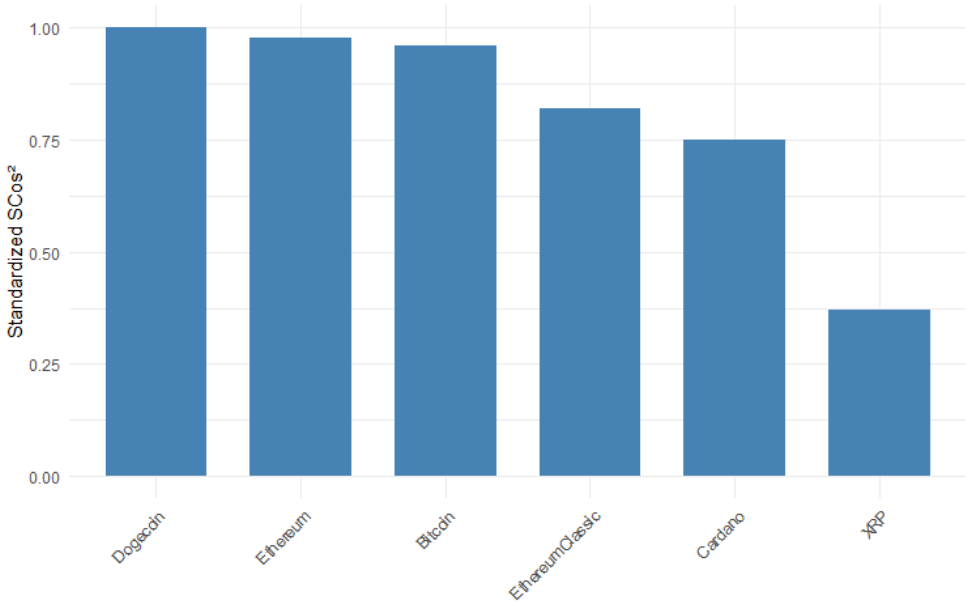


Figure 3.13: $SCos^2$ of Variables to Component 1 and 2 for Standardized Price Data (Covariance Matrix)

the covariance matrix. Dogecoin exhibits the highest Cos^2 value, approaching 1, indicating that its price variance is well captured by the first two components. Ethereum and Bitcoin follow closely, suggesting robust representation. Ethereum Classic shows moderate representation with a Cos^2 value around 0.75, while Cardano and XRP display lower values, particularly XRP, which falls below 0.5, implying that a significant portion of its variance lies outside the first two components.

The Figure 3.14 presents the standardized $SCos^2$ values of cryptocurrencies with respect to the first three principal components using the covariance matrix. The inclusion of the third component notably improves the representation for assets such as Cardano, Ethereum Classic and XRP. Dogecoin, Ethereum and Bitcoin exhibit high Cos^2 values approaching or reaching 1, suggesting that the first three principal components collectively

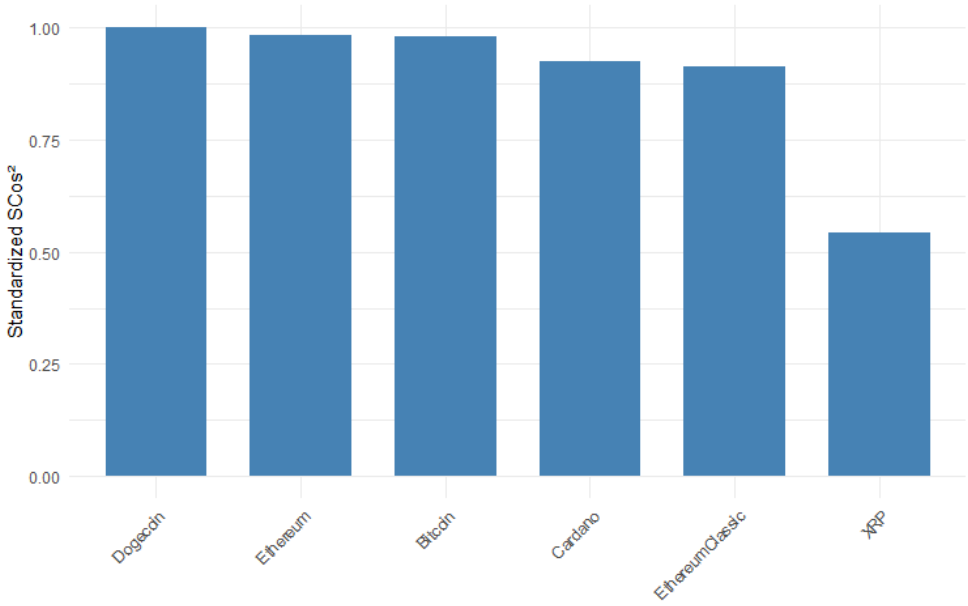


Figure 3.14: $SCos^2$ of Variables to Component 1 to 3 for Standardized Price Data (Covariance Matrix)

provide a strong representation of their variance. Cardano and Ethereum Classic also show robust representation, with values significantly improved compared to the two-component model. XRP, which had the lowest representation in the first two components, shows a noticeable increase to above 0.5, suggesting that the third component captures meaningful variance not accounted for previously. This enhancement confirms that the third component captures additional variance not accounted for by the first two components.

3.3 Factor Analysis

To further investigate the latent structure underlying cryptocurrency returns, we conduct factor analysis on the daily return data of six cryptocurrencies. Let $\mathbf{X}_{p\times 1}$ be a

random vector with mean vector μ and variance-covariance matrix Σ. The Factor model is expressed as

$$\mathbf{X}_{p\times 1} = \mu_{p\times 1} + \mathbf{L}_{p\times k}\mathbf{f}_{k\times 1} + \varepsilon_{p\times 1}$$

where μ is a vector of constants, $\mathbf{f}_{k\times 1}$ is a random vector of common factors, with $k < p$, $\mathbf{L}_{p\times k}$ is a matrix of unknown constants (factor loadings), $\varepsilon = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_p]^T$ is a random vector of specific factors. In the factor analysis, we assume the common factors and specific factors are uncorrelated $\mathrm{Cov}(\varepsilon, \mathbf{f}) = 0$, and

$$\begin{aligned} E(\mathbf{f}) &= 0 \\ E(\varepsilon) &= 0 \\ \mathrm{Var}(\mathbf{f}) &= \Delta, \quad \Delta \text{ is positive definite} \\ \mathrm{Var}(\varepsilon) &= \Psi = \mathrm{diag}(\psi_1, \ldots, \psi_p), \quad \psi_i > 0 \end{aligned}$$

So in the factor model, the variance of the observed variables is

$$\mathrm{Var}(\mathbf{x}) = \Sigma = \mathbf{L}\Delta\mathbf{L}' + \Psi$$

Without loss of generality, we can assume $\mathrm{Var}(\mathbf{f}) = \mathbf{I}_k$. Thus, the standard factor model is given by

$$\Sigma = \mathbf{L}\mathbf{L}' + \Psi$$

The objective is to determine the factor loading matrix $\mathbf{L}$ and the unique variance matrix Ψ such that the model assumptions are satisfied. In our analysis, we employ the maximum likelihood method to estimate $\mathbf{L}$ and Ψ. We assume that the observations $X_1, \ldots, X_n$ to be a random sample from a p-variate normal distribution with mean vector μ

and variance-covariance matrix Σ. Under this assumption, the maximum likelihood estimator (MLE) of Σ is given by $\hat{\Sigma} = \mathbf{S}$, the sample variance-covariance matrix. Furthermore,

$$(n-1)\mathbf{S} \sim \mathscr{W}_p(n-1,\Sigma), \quad \text{and} \quad E[\mathbf{S}] = \Sigma,$$

where $\mathscr{W}_p(n-1,\Sigma)$ denotes the Wishart distribution with $n-1$ degrees of freedom and scale matrix Σ. We utilize the probability density function of $\mathbf{S}$ to derive the log-likelihood function for $\mathbf{L}$ and Ψ, which is then maximized to obtain their MLEs (See details in Khattree and Naik (2000)).

In general, for the factor model $\Sigma = \mathbf{L}\mathbf{L}' + \Psi$, we require that the number of unknown parameters on the right-hand side be less than the number of distinct elements in the covariance matrix Σ, which appears on the left-hand side of the equation and has $\frac{p(p+1)}{2}$ distinct entries. On the right-hand side, $\mathbf{L}$ contains pk parameters. and Ψ is diagonal and contributes p parameters. Thus, the total number of parameters on the right-hand side is $pk+p$. For the factor model to be useful, we require

$$\frac{p(p+1)}{2} - (pk+p) > 0$$

Which simplifies to

$$p > 2k$$

In our case, we consider $p = 6$ cryptocurrencies. To keep the factor model effective , we choose the number of factors k such that

$$k < \frac{p}{2} = 3$$

and therefore select $k = 2$ factors. Additionally, based on the cumulative variance explained and the structure observed in prior principal component analysis results, we also test the model using the marginal case of $k = 3$ factors for the datasets.

To enhance interpretability, we apply varimax rotation, which redistributes the factor loadings so that each variable loads strongly on one factor and weakly on others. This rotation maintains orthogonality, ensuring that the extracted factors remain uncorrelated.

3.3.1 Factor Analysis of Daily Returns

Table 3.13: Summary Statistics for Factor Analysis with 2 Factors

	Bitcoin	**Ethereum**	**Ethereum Classic**	**XRP**	**Cardano**	**Dogecoin**
Uniquenesses	0.283	0.005	0.390	0.447	0.285	0.804
	Factor1	**Factor2**				
Bitcoin	0.646	0.547				
Ethereum	0.895	0.440				
Ethereum Classic	0.548	0.557				
XRP	0.414	0.618				
Cardano	0.506	0.678				
Dogecoin	0.195	0.398				
	Factor1	**Factor2**				
SS Loadings	1.983	1.802				
Proportion Var	0.331	0.300				
Cumulative Var	0.331	0.631				
Test of the hypothesis that 2 factors are sufficient: Chi-square statistic = 51.4, degrees of freedom = 4, p-value = 1.84×10^{-10}						

Table 3.13 summarizes the results of the factor analysis conducted on the daily returns of six cryptocurrencies using two factors and varimax rotation. The uniqueness values indicate the proportion of each asset's variance not explained by the common factors. Ethereum shows an extremely low uniqueness (0.005), suggesting that its behavior is almost entirely captured by the factor model. In contrast, Dogecoin has a high uniqueness (0.804), implying that it is largely independent of the extracted factors and may require additional components to be adequately modeled. For the remaining cryptocurrencies, more than half of their variances are explained by the two common factors, indicating that the model captures meaningful shared structure among them.

The rotated factor loadings reveal that most cryptocurrencies load moderately to strongly on both factors. Ethereum has the highest loading on Factor 1 (0.895), indicating that this factor may represent a dimension strongly influenced by Ethereum's daily returns. Cardano and XRP load more heavily on Factor 2 (0.678 and 0.618, respectively), suggesting that Factor 2 captures variation more aligned with these assets.

The two factors explain 33.1% and 30.0% of the total variance, respectively, with a cumulative variance of 63.1%. This indicates that the model captures a substantial portion of the variability in the data. To evaluate the adequacy of the two-factor model, a hypothesis test was performed. The chi-square statistic is 51.4 with 4 degrees of freedom, and the associated p-value is 1.84×10^{-10}. This highly significant result suggests that the two-factor model may not fully capture the underlying structure of the data, and that additional factors could improve model fit.

Table 3.14 presents the results of the factor analysis performed on the daily returns data using three factors and varimax rotation. Dogecoin and Ethereum exhibit extremely low uniqueness values (0.005 and 0.017, respectively), indicating that the common factors explain almost all the variance in these cryptocurrencies. While XRP and Ethereum

Table 3.14: Summary Statistics for Factor Analysis with 3 Factors

	Bitcoin	**Ethereum**	**Ethereum Classic**	**XRP**	**Cardano**	**Dogecoin**
Uniquenesses	0.285	0.017	0.396	0.431	0.271	0.005
	Factor1	**Factor2**	**Factor3**			
Bitcoin	0.598	0.564	0.200			
Ethereum	0.566	0.801	0.145			
Ethereum Classic	0.581	0.468	0.219			
XRP	0.681	0.296	0.133			
Cardano	0.746	0.379	0.172			
Dogecoin	0.174	0.136	0.973			
	Factor1	**Factor2**	**Factor3**			
SS Loadings	2.067	1.428	1.102			
Proportion Var	0.344	0.238	0.184			
Cumulative Var	0.344	0.582	0.766			

Classic show relatively higher uniqueness values (0.431 and 0.396, respectively), they are still within a reasonable range. Overall the uniqueness values are relatively low, suggesting that the three-factor model provides a good representation of the shared structure among these assets.

The rotated factor loadings reveal distinct associations between assets and factors. Factor 1 shows strong loadings across most cryptocurrencies, including Cardano, XRP, Ethereum Classic, Bitcoin, and Ethereum. This suggests that Factor 1 represents a general market trend that affects these cryptocurrencies similarly. Factor 2 has the strongest loading on Ethereum (0.801), indicating that Factor 2 captures variability that is particularly significant for Ethereum. Bitcoin (0.564) and Ethereum Classic (0.468) also have moderate loadings. Factor 3 is dominated by Dogecoin (0.973), suggesting that Factor 3 is primarily associated with Dogecoin. The other cryptocurrencies have much lower loadings, indicating that Factor 3 captures unique variability specific to Dogecoin.

The three factors explain 34.4%, 23.8%, and 18.4% of the total variance, respectively, with a cumulative variance of 76.6%. Compared to the two-factor model, this represents a substantial improvement in explanatory power. In terms of model adequacy, the test of the hypothesis that three factors are sufficient cannot be performed because the degrees of freedom is zero. This typically occurs when the number of parameters estimated equals the number of observed correlations, leaving no room for statistical testing.

3.3.2 Factor Analysis of Standardized Prices

Table 3.15: Summary Statistics for Factor Analysis with 2 Factors on Standardized Prices

	Bitcoin	**Ethereum**	**Ethereum Classic**	**XRP**	**Cardano**	**Dogecoin**
Uniquenesses	0.005	0.067	0.006	0.484	0.214	0.130
	Factor1	**Factor2**				
Bitcoin	0.334	0.940				
Ethereum	0.549	0.794				
Ethereum Classic	0.932	0.354				
XRP	0.663	0.276				
Cardano	0.730	0.503				
Dogecoin	0.769	0.527				
	Factor1	**Factor2**				
SS Loadings	2.846	2.247				
Proportion Var	0.474	0.375				
Cumulative Var	0.474	0.849				
Test of the hypothesis that 2 factors are sufficient: Chi-square statistic = 383.85, degrees of freedom = 4, p-value = 8.6×10^{-82}						

Table 3.15 presents the results of the factor analysis conducted on standardized daily prices of six cryptocurrencies using two factors and varimax rotation. The uniqueness values indicate that Bitcoin, Ethereum Classic, Ethereum, and Dogecoin exhibit low uniqueness values (0.005, 0.006, 0.067, and 0.130, respectively), suggesting that their price movements are almost entirely captured by the two-factor model. Cardano and XRP have moderately higher values (0.214 and 0.484, respectively).The uniqueness values are relatively low for all variables, indicating that the model effectively captures the shared structure among these assets.

The rotated factor loadings reveal that Factor 1 is strongly associated with Ethereum Classic, Dogecoin, Cardano, and XRP, suggesting a common market dynamic among these assets. Factor 2 is dominated by Bitcoin (0.940) and Ethereum (0.794), indicating a second dimension of variation likely tied to broader market leadership.

The two factors explain 47.4% and 37.5% of the total variance, respectively, with a cumulative variance of 84.9%. This represents a substantial improvement in explanatory power compared to previous models. To assess model adequacy, a hypothesis test was performed to evaluate whether two factors are sufficient. The chi-square statistic is 383.85 with 4 degrees of freedom, and the associated p-value is 8.6×10^{-82}. This highly significant result suggests that the two-factor model may not fully capture the underlying structure of the data, and additional factors could improve model fit.

The Table 3.16 presents the results of the factor analysis using three factors and varimax rotation. The uniqueness values in the three-factor model are uniformly low, with Bitcoin, XRP, and Ethereum Classic showing values below 0.03, and Dogecoin and Ethereum also well below or around 0.1. This indicates that the model captures the large portion of variance in these assets. Compared to the two-factor model (Table 3.15), the

Table 3.16: Summary Statistics for Factor Analysis with 3 Factors on Standardized Prices

	Bitcoin	**Ethereum**	**Ethereum Classic**	**XRP**	**Cardano**	**Dogecoin**
Uniquenesses	0.005	0.067	0.023	0.005	0.194	0.110
	Factor1	**Factor2**	**Factor3**			
Bitcoin	0.309	0.928	0.196			
Ethereum	0.510	0.777	0.263			
Ethereum Classic	0.841	0.331	0.399			
XRP	0.332	0.216	0.916			
Cardano	0.637	0.471	0.422			
Dogecoin	0.753	0.506	0.259			
	Factor1	**Factor2**	**Factor3**			
SS Loadings	2.147	2.099	1.350			
Proportion Var	0.358	0.350	0.225			
Cumulative Var	0.358	0.708	0.933			

reduction in uniqueness values, especially for XRP from 0.484 to 0.005, suggests a significantly improved fit.

The rotated factor loadings reveal more distinct structure. Factor 1 is primarily associated with Ethereum Classic, Dogecoin, and Cardano. Factor 2 is dominated by Bitcoin and Ethereum, while Factor 3 is largely associated with XRP. The three factors explain 35.8%, 35.0%, and 22.5% of the total variance, respectively, with a cumulative variance of 93.3%. This is a substantial improvement over the two-factor model, which explained 84.9% of the variance. The model fit test for the three-factor model cannot be performed due to zero degrees of freedom. In summary, The three-factor model offers better explanatory power and more distinct factor structure than the two-factor model, particularly in capturing the unique behavior of XRP and reducing unexplained variance across all assets.

3.4 Dynamic Factor Analysis

The Dynamic Factor Analysis (DFA) is a statistical methodology used to extract a small set of latent factors that explain the common dynamics of a larger collection of observed time series, see Stock and Watson (2016). In a DFA framework, each series is expressed as a weighted linear combination of the same latent processes, and these factors evolve through time. By using DFA, we can capture persistent temporal structures such as business cycles, long-run trends, seasonality, climate oscillations, and financial market regimes.

The primary distinction between classic factor analysis and Dynamic Factor Analysis lies in their treatment of time. Traditional factor analysis explains only cross sectional covariation at a single time point or implicitly ignores temporal ordering, and specifies no model for the evolution of latent factors. In contrast, DFA uses a full state-space representation, allowing the latent factors to follow a dynamic stochastic process. This structure enables DFA to model temporal behavior in a way that classic factor analysis cannot accommodate.

A dynamic factor model is formulated through two interconnected equations, which can be expressed as

$$x_t = C_0 f_t + e_t, \qquad e_t \sim \mathcal{N}(0, R) \tag{3.4}$$

$$f_t = \sum_{j=1}^{p} A_j f_{t-j} + u_t, \qquad u_t \sim \mathcal{N}(0, Q_0) \tag{3.5}$$

where equation 3.4 is the observation (measurement) equation, which specifies how the observed variables related to the latent factors, and equation 3.5 is the state (transition)

equation, which describes how the latent factors evolve over time according to a $VAR(p)$ process.

- x_t is an $n \times 1$ vector of observed series at time t, $(x_{1t}, \ldots, x_{nt})'$.
- f_t is an $r \times 1$ vector of latent factors at time t, $(f_{1t}, \ldots, f_{rt})'$ with $r \ll n$.
- C_0 is an $n \times r$ measurement (observation) matrix, also referred to as the factor loading matrix.
- A_j is an $r \times r$ state-transition matrix at lag j.
- Q_0 is an $r \times r$ state covariance matrix governing the process noise in the state equation.
- R is an $n \times n$ measurement (observation) covariance matrix, typically assumed diagonal under the assumption that covariance between the series is explained entirely by the latent factors.
- e_t is idiosyncratic noise in the measurement equation.
- u_t is process noise.

For our analysis, we employ the R package *dfms* to estimate the Dynamic Factor Model, see details in Krantz et al., 2023. We exclude the stablecoin Tether from the dataset because its unusually low variance can distort the scaling and disproportionately influence factor extraction. Before estimating the model, it is necessary to determine the appropriate number of latent factors. We use a scree plot to guide this choice by identifying a kink in the plot. As shown in Figure 3.15, the marginal contribution of factors 3 through 5 is minimal. Therefore, we estimate the model using two factors, which capture the dominant shared variation in the data.

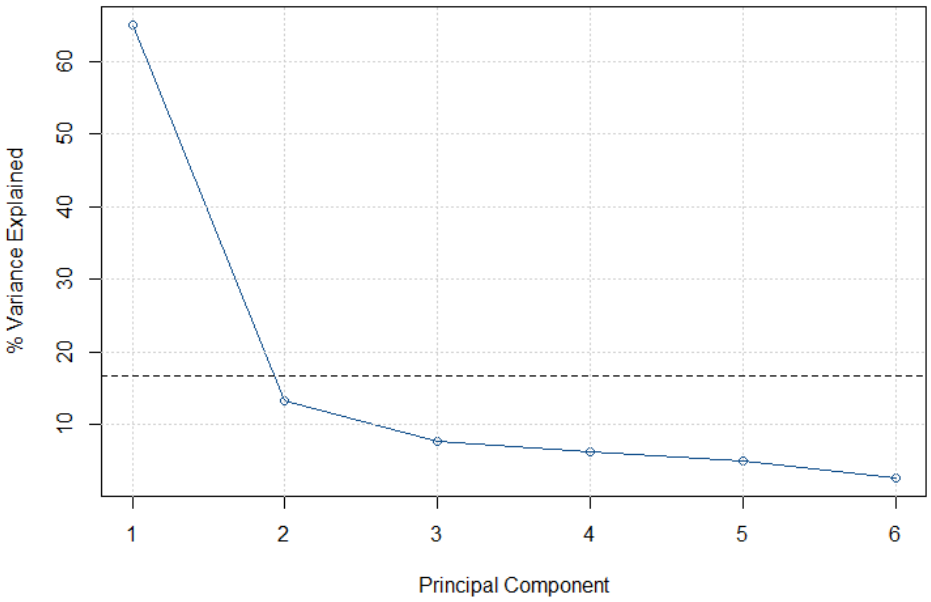


Figure 3.15: Scree Plot to Determine the Number of Factors

Next, we determine the lag order of the factor-VAR in the transition equation by evaluating standard information criteria and the final prediction error. We sequentially increase the lag length and compare model performance up to a $VAR(p)$ specification, with the results summarized in Table 3.17.

Based on the model selection results, we choose a lag order of $p = 3$ and retain $r = 2$ latent factors. Using these specifications, we fit the dynamic factor model to the daily returns of six major cryptocurrencies.

The estimated model decomposes the daily cryptocurrency returns into two economically interpretable latent forces. The first factor loads positively and fairly uniformly across all assets, indicating that it captures a broad, market-wide component of

Table 3.17: Lag Order Selection Criteria

	1	2	3	4	5	6	7	8	9	10
AIC	1.1210	1.1118	1.0989	1.0973	1.0998	1.0996	1.0970	1.0961	1.0959	1.0969
HQ	1.1264	1.1207	1.1114	1.1134	1.1195	1.1228	1.1238	1.1264	1.1298	1.1344
SC	1.1357	1.1363	1.1332	1.1415	1.1538	1.1633	1.1705	1.1794	1.1891	1.1999
FPE	3.0680	3.0398	3.0008	2.9962	3.0036	3.0029	2.9951	2.9924	2.9919	2.9950
Selected lags: AIC = 9, HQ = 3, SC = 3, FPE = 9										

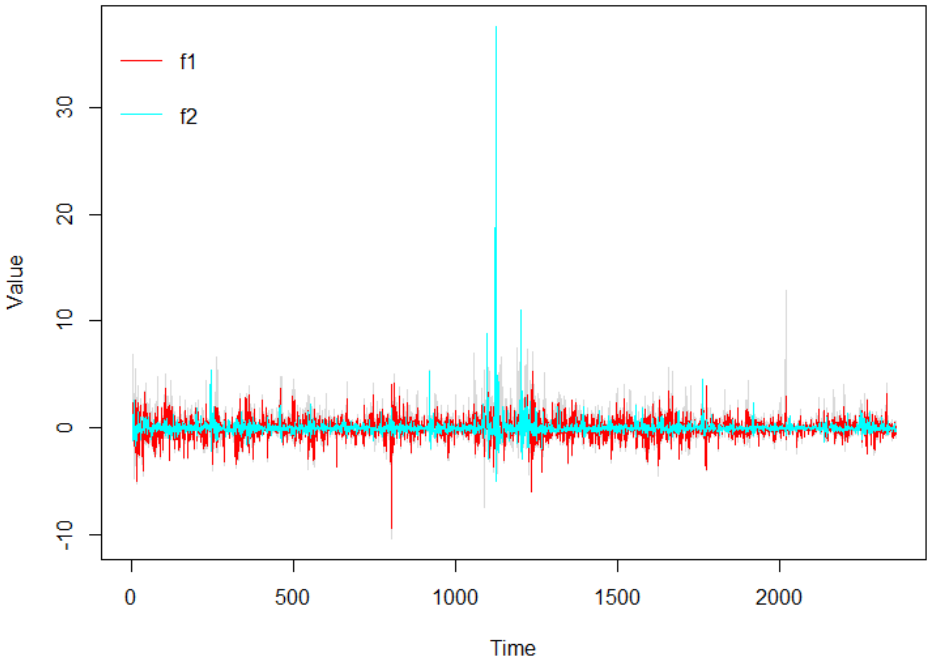


Figure 3.16: Standardized Series and Factor Estimates

cryptocurrency behavior. Its variance and shock variance, $\hat{\text{var}}(f_1) \approx 3.52$ and $\hat{Q}_{11} \approx 3.67$, suggest that this factor represents the primary high-volatility driver in the system.

Table 3.18: State-Space Model Matrices and Goodness of Fit

Factor Transition Matrix A						
	L1.f1	**L1.f2**	**L2.f1**	**L2.f2**	**L3.f1**	**L3.f2**
f1	-0.068788	0.17180	0.051309	-0.002175	0.03940	-0.0001701
f2	-0.006471	0.08203	-0.001437	-0.082133	0.01311	0.1145598
Factor Covariance Matrix $\text{cov}(F)$						
	f1	**f2**				
f1	3.5215	0.0953				
f2	0.0953	0.7681				
Factor Transition Error Covariance Matrix Q						
	u1	**u2**				
u1	3.6746	0.0284				
u2	0.0284	0.7717				
Observation Matrix C						
	f1	**f2**				
Bitcoin	0.4452	-0.0908				
Ethereum	0.4801	-0.1743				
Ethereum Classic	0.4045	-0.0388				
XRP	0.3640	-0.1032				
Cardano	0.4180	-0.0995				
Dogecoin	0.2584	0.9551				
Goodness of Fit (R^2)						
Bitcoin	0.7884					
Ethereum	0.9346					
Ethereum Classic	0.6459					
XRP	0.5314					
Cardano	0.6972					
Dogecoin	0.9999					

The second factor exhibits a distinctly different structure that it has an extremely large positive loading on Dogecoin and small negative loadings on the major cryptocurrencies. This configuration implies that the second factor reflects a form of

speculative rotation in which Dogecoin strongly outperforms while mainstream cryptocurrencies move more weakly or in the opposite direction.

The goodness-of-fit results support this interpretation. The R^2 values indicate that the two-factor model explains a substantial share of daily returns for several assets, notably Dogecoin (0.9999), Ethereum (0.9356), and Bitcoin (0.7884). The remaining cryptocurrencies display moderately strong fit as well. Moreover, both residual correlations and autocorrelations are small, suggesting that the factor structure successfully accounts for the shared dynamics across the series and leaves little systematic behavior unexplained.

3.5 Concluding Remarks

This chapter has presented a rigorous multivariate statistical analysis of major cryptocurrencies using Principal Component Analysis, Factor Analysis and Dynamic Factor Analysis. The PCA results demonstrate that a small number of principal components can effectively capture the majority of the variance in both daily returns and standardized price data, highlighting the first few components reveal strong co-movement among key assets. Ethereum and Bitcoin consistently emerged as dominant contributors to the first or second principal component, reflecting their central roles in the crypto market. Dogecoin and XRP exhibited distinct behaviors, often loading heavily on separate components, suggesting unique volatility patterns.

The analysis of both correlation and covariance matrices revealed that while correlation-based PCA provided balanced insights into co-movement patterns, covariance-based PCA highlighted the influence of high-variance assets such as Dogecoin. The use of Cos^2 metrics further clarified the representation quality of each cryptocurrency in the reduced component space.

The Factor Analysis further supports these findings, identifying latent structures that explain the interdependencies among cryptocurrencies. The three-factor model offered improved explanatory power over the two-factor model, especially in capturing the unique behavior of Dogecoin and XRP, and reducing unexplained variance. Low uniqueness values across most assets indicated that a limited number of common factors account for a substantial portion of the observed variability.

The Dynamic Factor Analysis extends these results by introducing a time-dependent representation of shared market dynamics. By modeling cryptocurrency returns through a state-space framework, the DFA uncovers two economically meaningful latent forces, a broad market-wide factor that drives co-ovement across major assets, and a second and more speculative factor that distinguishes Dogecoin from the rest of the market. This dynamic perspective complements the static PCA and Factor Analysis results, showing not only how cryptocurrencies are related contemporaneously but also how their shared drivers evolve over time.

These multivariate techniques provide a robust framework for understanding the structural relationships and latent dimensions of the cryptocurrency market, highlighting both shared market dynamics and asset-specific behaviors. The findings lay an analytical foundation for the predictive modeling and strategic applications developed in the subsequent chapters.

CHAPTER FOUR

CRYPTOCURRENCIES: PREDICTION OF DAILY RETURNS

4.1 Introduction

Building upon the descriptive and multivariate analyses presented in earlier chapters, this chapter advances the study into predictive modeling, with a focus on exploring the possibility of forecasting the daily returns of certain major cryptocurrencies. The objective is to develop some robust statistical models that leverage a diverse set of financial indicators, such as other cryptocurrencies, foreign exchange rates, various stock and commodity indices etc. to predict both the present-day and one-day-ahead returns.

Akyildirim et al. (2021) have explored the use of machine learning classification algorithms including support vector machines, logistic regression, artificial neural networks, and random forests, to predict the price direction using historical price data and technical indicators as model features, showing that there exists predictability of trends in prices to a certain degree in the cryptocurrency markets. Lamon et al. (2017) incorporated the sentiment analysis from news and social media into machine learning models and significantly enhanced the prediction of cryptocurrency price movements, using models like logistic regression and neural networks to classify directional shifts for Bitcoin, Ethereum, and Litecoin. Khedr et al. (2021) provided a comprehensive survey of cryptocurrency price prediction techniques, comparing traditional statistical methods with machine learning models, and highlighted that hybrid and deep learning approaches offer superior accuracy and adaptability in forecasting volatile crypto markets . Hasan et al. (2022) proposed an optimized cryptocurrency price prediction method using a convolutional neural network trained on sentiment-labeled Twitter data, achieving high accuracy in forecasting the prices of Bitcoin, Ethereum, Litecoin, and Monero by

correlating social media sentiment with market movements . However in most of these works, the question of how accurate these predictions may be on independent data remains unanswered.

The chapter instead employs a comprehensive modeling framework that integrates various classical regression techniques with modern regularization and shrinkage methods. These include full model estimation, submodel selection via AIC, BIC, and LASSO, and penalized regression approaches such as Ridge, Elastic Net, Adaptive LASSO, and SCAD. Post-shrinkage estimators are also introduced to enhance model stability and predictive accuracy.

Dimensionality reduction is achieved through Principal Component Analysis, which is applied to the predictor space to mitigate multicollinearity and improve interpretability. The resulting principal components are incorporated into regression models to assess their utility in forecasting cryptocurrency returns.

Model performance is evaluated using the bootstrap-based prediction error metrics, allowing for a rigorous comparison across methods and temporal horizons. The dual focus on present and future returns provides insights into the extent of dynamic relationships between the performances of cryptocurrencies and broader financial markets, offering valuable implications for investors.

4.2 Prediction Models on the Present (Same)-Day Data

To forecast cryptocurrency trends, we utilize a comprehensive set of leading financial indices as predictors. These include daily returns from twelve major foreign exchange (forex) pairs, five specific stock market indices, and three commodity indices. Specifically, these are:

- **Forex Indices**

  - EUR/USD: Euro vs. U.S. Dollar. (Conversion rate)
  - USD/JPY: U.S. Dollar vs. Japanese Yen.
  - GBP/USD: British Pound vs. U.S. Dollar.
  - CHF/USD: Swiss Franc vs. U.S. Dollar.
  - CAD/USD: Canadian Dollar vs. U.S. Dollar.
  - AUD/USD: Australian Dollar vs. U.S. Dollar.
  - NZD/USD: New Zealand Dollar vs. U.S. Dollar.
  - EUR/GBP: Euro vs. British Pound.
  - EUR/JPY: Euro vs. Japanese Yen.
  - GBP/JPY: British Pound vs. Japanese Yen.
  - AUD/JPY: Australian Dollar vs. Japanese Yen.
  - CHF/JPY: Swiss Franc vs. Japanese Yen.

- **Stock Market Indices**
  - S&P 500: Standard & Poor's 500 Index which tracks the performance of the 500 largest publicly traded companies in the U.S. Examples of Companies: Apple, Microsoft, Amazon, Tesla, Google (Alphabet).
  - DJIA: Dow Jones Industrial Average Index which measures the performance of 30 large U.S. companies. Such as Coca-Cola, Boeing, Goldman Sachs, McDonald's.
  - NASDAQ: Nasdaq Composite Index which tracks more than 3,000 companies listed on the Nasdaq exchange, with a focus on technology stocks. Examples of Companies: Apple, NVIDIA, Meta, Tesla, Amazon.

  - Nasdaq 100: Nasdaq 100 Index which is made up of 100 of the largest non-financial companies listed on the Nasdaq Stock Market.
  - Russell 2000: Russell 2000 Index which measures the performance of approximately 2,000 small-cap companies in the United States, it is a subset of the Russell 3000 Index.

- **Commodity Indices**
  - DJOilGas: Dow Jones US Oil Gas Index which measures the performance of companies in the oil and gas industry within the United States.
  - XAU/USD: Spot Gold price quoted in U.S. dollars per ounce. This tracks the real-time market value of gold.
  - DJGoldMining: Dow Jones US Gold Mining Index is a stock market index that measures the performance of U.S. companies involved in the gold mining industry.

We retrieve the closing prices or values for above 20 different financial indices from January 1, 2018, to June 17, 2024 from the Investing.com website. The Table 4.1 displays the first five rows of the combined dataset. Due to the nature of stock market indices, weekends and holidays will lead to missing values for those specific dates.

Table 4.1: Financial Data

Date	**EUR_USD**	**USD_JPY**	**GBP_USD**	**CHF_USD**	**CAD_USD**
2018-01-01	1.2010	112.64	1.3503	1.0262	0.7972
2018-01-02	1.2059	112.28	1.3588	1.0292	0.7992
2018-01-03	1.2014	112.49	1.3516	1.0235	0.7978

2018-01-04	1.2068	112.74	1.3553	1.0261	0.8009
2018-01-05	1.2030	113.06	1.3571	1.0254	0.8058
Date	**AUD_USD**	**NZD_USD**	**EUR_GBP**	**EUR_JPY**	**GBP_JPY**
2018-01-01	0.7805	0.7106	0.8894	135.30	152.12
2018-01-02	0.7830	0.7105	0.8875	135.42	152.59
2018-01-03	0.7836	0.7092	0.8890	135.18	152.06
2018-01-04	0.7864	0.7155	0.8904	136.06	152.81
2018-01-05	0.7864	0.7169	0.8865	136.03	153.46
Date	**AUD_JPY**	**CHF_JPY**	**S&P 500**	**DowJones**	**NASDAQ**
2018-01-01	87.92	115.60	-	-	-
2018-01-02	87.92	115.58	2695.8	24824.01	7006.90
2018-01-03	88.14	115.15	2713.1	24922.68	7065.53
2018-01-04	88.67	115.69	2724.0	25075.13	7077.91
2018-01-05	88.92	115.95	2743.2	25295.87	7136.56
Date	**XAU_USD**	**Nasdaq100**	**DJOilGas**	**DJGoldMining**	**Russell2000**
2018-01-01	1306.86	-	-	-	-
2018-01-02	1318.14	6511.34	604.72	84.41	1550.01
2018-01-03	1312.84	6575.80	613.81	83.77	1552.58
2018-01-04	1322.97	6584.58	617.42	84.59	1555.72
2018-01-05	1320.21	6653.29	616.96	84.67	1560.01

After merging the financial index dataset with the cryptocurrency dataset and removing missing values attributable to stock market closures, the number of observations categorized by year, month, and weekday is presented in Table 4.2.

Table 4.2: Number of Observations by Year, Month, and Weekday

Year	**Freq.**	**Month**	**Freq.**	**Weekday**	**Freq.**
2018	241	01	130	Monday	291
2019	243	02	127	Tuesday	298
2020	244	03	153	Wednesday	330
2021	243	04	138	Thursday	326
2022	242	05	143	Friday	319
2023	240	06	135		
2024	111	07	120		
		08	134		
		09	116		
		10	132		
		11	118		
		12	118		

For each financial index, the daily change z_t is computed using the formula

$$z_t = \frac{v_t - v_{t-1}}{v_{t-1}}$$

where v_t represents the closing value on day t. Subsequently, we construct prediction models in which the daily return of each cryptocurrency serves as the response variable, while the set of twenty financial indices described above along with the returns of other cryptocurrencies are employed as the explanatory variables. For example, the full model for predicting the daily return of Bitcoin is given by,

$$
\begin{aligned}
Y \; &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \\
&+ \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} \\
&+ \beta_{15} X_{15} + \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} \\
&+ \beta_{22} X_{22} + \beta_{23} X_{23} + \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \varepsilon
\end{aligned}
\tag{4.1}
$$

Where

- Y: Daily return of Bitcoin
- X_1: Daily return of EUR/USD exchange rate
- X_2: Daily return of USD/JPY exchange rate
- X_3: Daily return of GBP/USD exchange rate
- X_4: Daily return of CHF/USD exchange rate
- X_5: Daily return of CAD/USD exchange rate
- X_6: Daily return of AUD/USD exchange rate
- X_7: Daily return of NZD/USD exchange rate
- X_8: Daily return of EUR/GBP exchange rate
- X_9: Daily return of EUR/JPY exchange rate
- X_{10}: Daily return of GBP/JPY exchange rate
- X_{11}: Daily return of AUD/JPY exchange rate
- X_{12}: Daily return of CHF/JPY exchange rate

- X_{13}: Daily return of S&P 500 Index
- X_{14}: Daily return of Dow Jones Index
- X_{15}: Daily return of NASDAQ Index
- X_{16}: Daily return of Russell 2000 Index
- X_{17}: Daily return of Gold price in USD (XAU/USD)
- X_{18}: Daily return of NASDAQ 100 Index
- X_{19}: Daily return of Dow Jones Oil and Gas Index
- X_{20}: Daily return of Dow Jones Gold Mining Index
- X_{21}: Daily return of Ethereum
- X_{22}: Daily return of Ethereum Classic
- X_{23}: Daily return of Tether
- X_{24}: Daily return of XRP
- X_{25}: Daily return of Cardano
- X_{26}: Daily return of Dogecoin
- $\beta_0, \ldots, \beta_{26}$: Unknown regression parameters
- ε: Random error term

We assume the error term is randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{1564\times1} = \mathbf{X}_{1564\times27}\beta_{27\times1} + \varepsilon_{1564\times1}, \;\; Rank(\mathbf{X}) = 27 < 1564 \tag{4.2}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin daily returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 27$ predictors including the intercept, β is the vector of unknown regression parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

The modeling framework is grounded in the estimation strategies and shrinkage theory discussed in Chapter 1. Specifically, we apply a combination of classical and modern regression techniques, including full model estimation, submodel selection via AIC, BIC, and LASSO, and penalized methods such as Ridge, LASSO, Elastic Net, Adaptive LASSO, and Smoothly Clipped Absolute Deviation. Post-shrinkage and positive shrinkage estimators are also employed to improve robustness and predictive accuracy. To evaluate the model performance, we implement a bootstrap-based procedure with 250 iterations. Each iteration follows these steps:

1. Randomly split the bootstrap sample into training and testing sets in a $3/4 : 1/4$ ratio.

2. Fit the full model on the training data and estimate coefficients for all predictors.

3. Fit three submodels, selected using AIC, BIC, and LASSO, and estimate their respective coefficients.

4. Conduct likelihood ratio tests comparing the full model with each submodel. Using the resulting test statistics, compute shrinkage and positive shrinkage estimators as described in Chapter 1.

5. Fit five penalized models (LASSO, ENET, ALASSO, SCAD, and Ridge) on the training data and store their coefficients.

6. Using the test data, compute prediction errors for:

   (a) The full model

   (b) The three submodels

   (c) The three shrinkage estimator models

   (d) The three positive shrinkage estimator models

   (e) The five penalized models

7. Aggregate the prediction errors across all bootstrap samples and compute their averages and standard deviations.

This comprehensive framework allows for a rigorous comparison of model performances and provides insights into the predictive relationships between cryptocurrency returns and broader financial market indicators. One may expect high multicollinearity among some of the variables. We expect this to be a case especially among various FOREX pairs. Additionally, indexes such as Nasdaq and Nasdaq 100 may be highly correlated because the latter consists of a subset of stocks from the former.

4.2.1 A Model for Bitcoin

To evaluate the predictive performance for Bitcoin's daily returns, we begin by fitting a full linear regression model using all twenty financial indices and the returns of other cryptocurrencies as predictors. As shown in Table 4.3, the full model explains a reasonably large portion of the variance in Bitcoin returns, with an R^2 value of 0.685, indicating a strong explanatory power. Among the predictors, daily returns of NASDAQ,

Table 4.3: Coefficients and Model Statistics for the Full Model of Bitcoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.035615	0.061763	0.577	0.5643	-
EUR_USD_return	0.329651	4.314137	0.076	0.9391	1046.077
USD_JPY_return	-2.841820	5.943997	-0.478	0.6326	2719.855
GBP_USD_return	-1.023811	3.155591	-0.324	0.7456	898.570
CHF_USD_return	-2.145481	6.556212	-0.327	0.7435	2422.214
CAD_USD_return	-0.076483	0.224951	-0.340	0.7339	2.615
AUD_USD_return	0.248482	2.918469	0.085	0.9322	983.835
NZD_USD_return	-0.042834	0.204359	-0.210	0.8340	4.581
EUR_GBP_return	3.485431	3.735682	0.933	0.3510	672.529
EUR_JPY_return	-3.695613	5.361308	-0.689	0.4907	2148.460
GBP_JPY_return	4.465170	4.656873	0.959	0.3378	2226.654
AUD_JPY_return	-0.187049	2.927213	-0.064	0.9491	1095.165
CHF_JPY_return	2.138964	6.566157	0.326	0.7447	2693.918
SP500_return	-0.652113	0.426719	-1.528	0.1267	80.202
DowJones_return	0.266627	0.252012	1.058	0.2902	27.150
NASDAQ_return	1.611100	0.668976	2.408	0.0161	278.881
Russell2000_return	-0.131234	0.128599	-1.020	0.3077	11.817
XAU_USD_return	0.028782	0.102229	0.282	0.7783	2.261
Nasdaq100_return	-1.033646	0.566789	-1.824	0.0684	212.884
DJOilGas_return	0.013229	0.047743	0.277	0.7818	2.773
DJGoldMining_return	0.062522	0.041367	1.511	0.1309	1.965
Ethereum_return	0.455265	0.019885	22.895	$< 2 \times 10^{-16}$	3.117
EthereumClassic_return	0.026235	0.013709	1.914	0.0558	2.279
Tether_return	0.137592	0.169440	0.812	0.4169	1.062
XRP_return	0.011084	0.012927	0.857	0.3913	2.012
Cardano_return	0.124761	0.015731	7.931	4.16×10^{-15}	2.708
Dogecoin_return	0.031063	0.005615	5.532	3.71×10^{-8}	1.196
Model Statistics					
Residual standard error: 2.397 on 1537 degrees of freedom					
Multiple R-squared: 0.685					
Adjusted R-squared: 0.6796					
F-statistic: 128.5 on 26 and 1537 DF					
p-value: $< 2.2 \times 10^{-16}$					

Ethereum, Cardano, and Dogecoin are statistically significant, suggesting their influential roles in Bitcoin price movement via this conditional model. However, the presence of large variance inflation factors (VIF) for several predictors indicates multicollinearity, which may distort the interpretation of individual coefficients and their signs. Therefore, interpreting the estimated effects should be made with caution. The overall model is statistically significant, as indicated by its large F-statistic and very small p-value, notwithstanding the fact that number of observations $(n = 1564)$ is large.

Table 4.4: Selected Variables for Submodels on Bitcoin

Criterion	**Variables Selected**
BIC	DJGoldMining, Ethereum, Cardano, Dogecoin
AIC	SP500, NASDAQ, Nasdaq100, DJGoldMining, Ethereum, Ethereum Classic, Cardano, Dogecoin
LASSO	NASDAQ, DJGoldMining, Ethereum, Ethereum Classic, XRP, Cardano, Dogecoin

To enhance model interpretability and reduce overfitting, we apply submodel selection techniques using AIC, BIC, and LASSO. The variables selected by each criterion are summarized in Table 4.4. These submodels serve as the basis for constructing shrinkage and positive shrinkage estimators, as discussed in Chapter 1.

Model performance is assessed using a bootstrap procedure with 250 iterations. The average and standard deviation of prediction errors across bootstrap samples are reported in Table 4.5. The full model has an average prediction error of approximately 6.06 with a standard deviation of approximately 1.26. This serves as the benchmark for comparing other models. Among all models, the AIC-selected submodel yields the lowest

Table 4.5: Average and Standard Deviation of Prediction Errors for Bitcoin Daily Return

Estimator	PE Average	PE s.d
Full Model	6.055481	1.263666
Submodel BIC	5.974263	1.274064
Shrinkage Estimator BIC	5.974864	1.276666
Positive Shrinkage Estimator BIC	5.974740	1.276601
Submodel AIC	5.938739	1.209824
Shrinkage Estimator AIC	5.966357	1.213152
Positive Shrinkage Estimator AIC	5.966804	1.221443
Submodel LASSO	6.005206	1.282170
Shrinkage Estimator LASSO	5.998524	1.280712
Positive Shrinkage Estimator LASSO	5.998370	1.280982
LASSO	6.022855	1.222497
ENET	6.014935	1.213455
ALASSO	6.040413	1.264689
SCAD	6.034406	1.366890
RIDGE	6.072061	1.172640

average prediction error and standard deviation, indicating superior predictive performance. BIC-based submodels and their shrinkage variants perform slightly better than the full model, while LASSO-based models show comparable results with modest improvements in their shrinkage forms. RIDGE model has the highest average prediction error among all models, while SCAD has the highest standard deviation, indicating less consistent performance. This analysis demonstrates the effectiveness of model selection and regularization techniques in improving the accuracy and stability of cryptocurrency's return predictions.

4.2.2 A Model for Ethereum

We fit the similar model for Ethereum now. The Bitcoin return is now taken as one of the explanatory variable. The full model estimation for Ethereum's daily return is

Table 4.6: Coefficients and Model Statistics for the Full Model of Ethereum

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.025377	0.068419	0.371	0.7108	-
EUR_USD_return	-0.960094	4.778745	-0.201	0.8408	1046.054
USD_JPY_return	12.922815	6.576440	1.965	0.0496	2713.443
GBP_USD_return	2.714226	3.494904	0.777	0.4375	898.279
CHF_USD_return	12.706552	7.255379	1.751	0.0801	2417.558
CAD_USD_return	0.473591	0.248896	1.903	0.0573	2.610
AUD_USD_return	-2.296045	3.232285	-0.710	0.4776	983.517
NZD_USD_return	0.372114	0.226174	1.645	0.1001	4.573
EUR_GBP_return	-2.966365	4.138521	-0.717	0.4736	672.685
EUR_JPY_return	3.788858	5.938889	0.638	0.5236	2148.556
GBP_JPY_return	-5.489496	5.158092	-1.064	0.2874	2226.345
AUD_JPY_return	2.010794	3.242092	0.620	0.5352	1094.894
CHF_JPY_return	-13.191459	7.265841	-1.816	0.0696	2688.339
SP500_return	-0.742270	0.472660	-1.570	0.1165	80.195
DowJones_return	0.494607	0.278972	1.773	0.0764	27.114
NASDAQ_return	0.539705	0.742298	0.727	0.4673	279.837
Russell2000_return	0.013233	0.142498	0.093	0.9260	11.825
XAU_USD_return	0.062434	0.113232	0.551	0.5815	2.260
Nasdaq100_return	-0.202158	0.628493	-0.322	0.7478	213.330
DJOilGas_return	-0.053092	0.052869	-1.004	0.3154	2.771
DJGoldMining_return	-0.025152	0.045852	-0.549	0.5834	1.968
Bitcoin_return	0.558616	0.024399	22.895	$< 2 \times 10^{-16}$	2.367
EthereumClassic_return	0.223348	0.014096	15.845	$< 2 \times 10^{-16}$	1.964
Tether_return	-0.127793	0.187702	-0.681	0.4961	1.062
XRP_return	0.096985	0.014108	6.874	9.02×10^{-12}	1.953
Cardano_return	0.143841	0.017396	8.269	2.90×10^{-16}	2.698
Dogecoin_return	-0.011766	0.006274	-1.875	0.0609	1.217
Model Statistics					
Residual standard error: 2.655 on 1537 degrees of freedom					
Multiple R-squared: 0.7608					
Adjusted R-squared: 0.7567					
F-statistic: 188 on 26 and 1537 DF					
p-value: $< 2.2 \times 10^{-16}$					

presented in Table 4.6. This model explains approximately 76.08% of the variance in Ethereum's daily returns, as indicated by the multiple R^2 value. The overall model is statistically significant, with an F-statistic of 188 on (26, 1537) degree of freedom and a p-value less than 2×10^{-16}. Among the predictors, daily returns of Bitcoin, Ethereum Classic, XRP, and Cardano exhibit strong positive associations with Ethereum's return, while USD/JPY, CHF/JYP and CAD/USD returns also show marginal significance. These results suggest that Ethereum's price dynamics are influenced not only by other cryptocurrencies but also by a few select macroeconomic indicators. High VIF values observed for several predictors suggest the presence of multicollinearity, which may undermine the reliability of individual coefficient estimates and their signs, warranting cautious interpretation.

Table 4.7: Selected Variables for Submodels on Ethereum

Criterion	**Variables Selected**
BIC	CAD_USD, Bitcoin, Ethereum Classic, XRP, Cardano
AIC	USD_JPY, CHF_USD, CAD_USD, CHF_JPY, SP500, DowJones, NASDAQ, Bitcoin, Ethereum Classic, XRP, Cardano, Dogecoin
LASSO	USD_JPY, CHF_USD, CAD_USD, AUD_USD, NZD_USD, EUR_GBP, GBP_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, DJOilGas, DJGoldMining, Bitcoin, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

The selected variables for each submodel selection criterion are summarized in Table 4.7. Additionally, we evaluate the predictive accuracy of various estimation strategies, including shrinkage and penalized methods, using bootstrap-based prediction error metrics. As shown in Table 4.8, models incorporating shrinkage estimators, particularly those based on BIC, demonstrate improved predictive performance compared to the full model. Regularization methods such as LASSO and Ridge also yield competitive results, while Elastic Net exhibits the lowest prediction error among penalized approaches.

Table 4.8: Average and Standard Deviation of Prediction Errors for Ethereum Daily Return

Estimator	**PE Average**	**PE s.d**
Full Model	7.250630	1.510995
Submodel BIC	7.225552	1.517791
Shrinkage Estimator BIC	7.173869	1.514233
Positive Shrinkage Estimator BIC	7.173755	1.514117
Submodel AIC	7.200506	1.522865
Shrinkage Estimator AIC	7.201924	1.516492
Positive Shrinkage Estimator AIC	7.199984	1.516783
Submodel LASSO	7.249310	1.505100
Shrinkage Estimator LASSO	7.243777	1.509705
Positive Shrinkage Estimator LASSO	7.242555	1.509524
LASSO	7.234660	1.507198
ENET	7.225176	1.503732
ALASSO	7.248398	1.503271
SCAD	7.267903	1.526932
RIDGE	7.232402	1.468275

Table 4.9: Coefficients and Model Statistics for the Full Model of Ethereum Classic

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.02191	0.11479	-0.191	0.84868	-
EUR_USD_return	-0.87686	8.01754	-0.109	0.91293	1046.073
USD_JPY_return	-13.82810	11.04173	-1.252	0.21063	2717.487
GBP_USD_return	4.49978	5.86355	0.767	0.44295	898.288
CHF_USD_return	-12.39884	12.18063	-1.018	0.30888	2420.751
CAD_USD_return	-0.01112	0.41807	-0.027	0.97878	2.616
AUD_USD_return	-4.10208	5.42279	-0.756	0.44949	983.474
NZD_USD_return	-0.61591	0.37947	-1.623	0.10478	4.573
EUR_GBP_return	-10.57125	6.93926	-1.523	0.12787	671.896
EUR_JPY_return	11.66434	9.96075	1.171	0.24177	2147.209
GBP_JPY_return	-15.36435	8.64822	-1.777	0.07583	2223.420
AUD_JPY_return	4.37625	5.43890	0.805	0.42116	1094.707
CHF_JPY_return	12.85991	12.19880	1.054	0.29196	2692.158
SP500_return	2.42343	0.79122	3.063	0.00223	79.837
DowJones_return	-1.26495	0.46741	-2.706	0.00688	27.041
NASDAQ_return	-0.77553	1.24544	-0.623	0.53358	279.862
Russell2000_return	-0.24657	0.23899	-1.032	0.30236	11.817
XAU_USD_return	0.18167	0.18993	0.956	0.33898	2.259
Nasdaq100_return	-0.01505	1.05448	-0.014	0.98861	213.344
DJOilGas_return	0.06248	0.08872	0.704	0.48134	2.772
DJGoldMining_return	-0.01945	0.07693	-0.253	0.80043	1.968
Bitcoin_return	0.09061	0.04735	1.914	0.05584	3.167
Ethereum_return	0.62868	0.03968	15.845	$< 2 \times 10^{-16}$	3.593
Tether_return	-0.09304	0.31495	-0.295	0.76771	1.062
XRP_return	0.04137	0.02401	1.723	0.08505	2.010
Cardano_return	0.15122	0.02958	5.113	3.57×10^{-7}	2.771
Dogecoin_return	0.06378	0.01041	6.126	1.15×10^{-9}	1.191
Model Statistics					
Residual standard error: 4.454 on 1537 degrees of freedom					
Multiple R-squared: 0.5623					
Adjusted R-squared: 0.5549					
F-statistic: 75.95 on 26 and 1537 DF					
p-value: $< 2.2 \times 10^{-16}$					

4.2.3 A Model for Ethereum Classic

The full model estimation with similar obvious changes as done in previous subsection for Ethereum Classic's daily return is summarized in Table 4.9. The model accounts for approximately 56.23% of the variability in Ethereum Classic returns. The overall model is statistically significant, as indicated by an F-statistic of 75.95 on 26 and 1537 DF and a p-value less than 2×10^{-16}. Among the predictors, the returns of Ethereum, Cardano, Dogecoin, the S&P 500, and the Dow Jones exhibit statistically significant relationships with Ethereum Classic. However, the presence of high VIF values suggests potential multicollinearity among the predictors. These findings suggest that Ethereum Classic's price dynamics are influenced by both cryptocurrency-specific factors and broader market indicators.

Table 4.10: Selected Variables for Submodels on Ethereum Classic

Criterion	**Variables Selected**
BIC	Ethereum, Cardano, Dogecoin
AIC	USD_JPY, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, SP500, DowJones, NASDAQ, Bitcoin, Ethereum, XRP, Cardano, Dogecoin
LASSO	EUR_USD, USD_JPY, CHF_USD, AUD_USD, NZD_USD, GBP_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Tether, XRP, Cardano, Dogecoin

The selected variables for each submodel selection method are listed in Table 4.10. As shown in Table 4.11, the shrinkage estimators based on BIC yield the lowest average

prediction error and demonstrate consistent performance across bootstrap samples. Ridge and Elastic Net regressions also perform well, while SCAD exhibits higher variability in prediction accuracy.

Table 4.11: Average and Standard Deviation of Prediction Errors for Ethereum Classic Daily Return

Estimator	**PE Average**	**PE s.d**
Full Model	20.81198	6.302590
Submodel BIC	20.70955	6.295733
Shrinkage Estimator BIC	20.57093	6.299083
Positive Shrinkage Estimator BIC	20.57093	6.299083
Submodel AIC	20.63642	6.256971
Shrinkage Estimator AIC	20.65916	6.280452
Positive Shrinkage Estimator AIC	20.65568	6.277612
Submodel LASSO	20.74435	6.306430
Shrinkage Estimator LASSO	20.76855	6.312543
Positive Shrinkage Estimator LASSO	20.76583	6.310252
LASSO	20.73753	6.321986
ENET	20.71220	6.318565
ALASSO	20.75492	6.334878
SCAD	20.89417	6.453250
RIDGE	20.63337	6.151357

4.2.4 A Model for Tether

The full model estimation for Tether's daily return is presented in Table 4.12. Unlike other cryptocurrencies analyzed and consistent with its design as a stablecoin, Tether exhibits minimal variability in its daily returns. The model explains only a small portion of the variance ($R^2 = 0.0587$), indicating limited explanatory power. Nonetheless,

Table 4.12: Coefficients and Model Statistics for the Full Model of Tether

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.0030594	0.0092963	0.329	0.74213	-
EUR_USD_return	-0.1252595	0.6492978	-0.193	0.84705	1046.056
USD_JPY_return	-0.1357838	0.8946685	-0.152	0.87939	2720.219
GBP_USD_return	0.1581570	0.4749353	0.333	0.73917	898.567
CHF_USD_return	1.0471131	0.9864237	1.062	0.28862	2420.608
CAD_USD_return	0.0592015	0.0338241	1.750	0.08027	2.610
AUD_USD_return	-1.1555482	0.4382588	-2.637	0.00846	979.410
NZD_USD_return	-0.0151510	0.0307554	-0.493	0.62234	4.580
EUR_GBP_return	1.1274232	0.5616669	2.007	0.04489	671.151
EUR_JPY_return	-0.9662216	0.8066585	-1.198	0.23118	2147.120
GBP_JPY_return	0.9861970	0.7006466	1.408	0.15947	2225.117
AUD_JPY_return	1.1564005	0.4395758	2.631	0.00861	1090.259
CHF_JPY_return	-1.0451818	0.9879220	-1.058	0.29024	2692.144
SP500_return	-0.2759633	0.0638860	-4.320	1.66×10^{-5}	79.360
DowJones_return	0.0625646	0.0379097	1.650	0.09907	27.121
NASDAQ_return	0.0962766	0.1008449	0.955	0.33988	279.767
Russell2000_return	0.0176160	0.0193563	0.910	0.36292	11.818
XAU_USD_return	-0.0374762	0.0153568	-2.440	0.01478	2.252
Nasdaq100_return	0.0208460	0.0853958	0.244	0.80718	213.336
DJOilGas_return	0.0062310	0.0071840	0.867	0.38589	2.772
DJGoldMining_return	0.0164046	0.0062165	2.639	0.00840	1.959
Bitcoin_return	0.0031168	0.0038382	0.812	0.41689	3.173
Ethereum_return	-0.0023592	0.0034652	-0.681	0.49608	4.179
EthereumClassic_return	-0.0006102	0.0020657	-0.295	0.76771	2.285
XRP_return	-0.0017172	0.0019456	-0.883	0.37760	2.012
Cardano_return	0.0003833	0.0024156	0.159	0.87393	2.818
Dogecoin_return	0.0008914	0.0008531	1.045	0.29621	1.219
Model Statistics					
Residual standard error: 0.3607 on 1537 degrees of freedom					
Multiple R-squared: 0.05874					
Adjusted R-squared: 0.04282					
F-statistic: 3.689 on 26 and 1537 DF					
p-value: 1.367×10^{-9}					

due to large sample, the overall model is statistically significant (F-statistic = 3.689, p-value $< 1.37 \times 10^{-9}$). Among the predictors, despite high variance inflation factor showing multicollinearity, several financial indices, including S&P 500, Dow Jones Gold Mining Index, XAU/USD, AUD/JPY and AUD/USD, show significance. However, most cryptocurrency returns do not exhibit strong associations with Tether's daily return. These observations align well with its intended stability and low volatility.

Table 4.13: Selected Variables for Submodels on Tether

Criterion	**Variables Selected**
BIC	SP500, NASDAQ, CAD_USD
AIC	CHF_USD, CAD_USD, AUD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, DJGoldMining
LASSO	No variables selected

Various Submodel selections using AIC, BIC, and LASSO yield more parsimonious models, with selected variables summarized in Table 4.13. Notably, the LASSO method does not select any predictors, reinforcing the limited signal in Tether's return data.

As shown in the Table 4.14, the submodel selected via AIC achieves the lowest average prediction error and standard deviation, indicating superior performance in terms of both accuracy and consistency. The full Model also performs well, while the submodel via BIC shows the poorest performance. Shrinkage models generally offer moderate

improvements, while penalized methods such as LASSO and ENET yield comparable results.

Table 4.14: Average and Standard Deviation of Prediction Errors for Tether Daily Return

Estimator	**PE Average**	**PE s.d**
Full Model	0.1358201	0.03847833
Submodel BIC	0.2073139	0.09363704
Shrinkage Estimator BIC	0.1409844	0.04786311
Positive Shrinkage Estimator BIC	0.1409564	0.04783200
Submodel AIC	0.1339601	0.03810189
Shrinkage Estimator AIC	0.1345530	0.03833500
Positive Shrinkage Estimator AIC	0.1345375	0.03832691
Submodel LASSO	-	-
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	0.1358738	0.04711110
ENET	0.1359511	0.04733080
ALASSO	0.1365595	0.04643792
SCAD	0.1359888	0.04099468
RIDGE	0.1360254	0.03986991

4.2.5 A Model for XRP

The full model estimation for XRP's daily return is presented in Table 4.15. This model explains approximately 50.33% of the variability in XRP returns (Multiple R^2 of 0.5033) and is statistically significant overall, as indicated by an F-statistic of 59.91 and a p-value less than 2.2×10^{-16}. Among the predictors, Ethereum and Cardano returns show strong statistical associations with XRP returns, while Ethereum Classic and XAU/USD exhibit marginal significance. The presence of high VIF values for some predictors

Table 4.15: Coefficients and Model Statistics for the Full Model of XRP

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.008070	0.121848	0.066	0.9472	-
EUR_USD_return	-3.801472	8.509648	-0.447	0.6551	1045.946
USD_JPY_return	-4.471062	11.725609	-0.381	0.7030	2720.003
GBP_USD_return	0.745407	6.224990	0.120	0.9047	898.623
CHF_USD_return	-5.796275	12.932570	-0.448	0.6541	2422.066
CAD_USD_return	-0.726598	0.443374	-1.639	0.1015	2.611
AUD_USD_return	5.084125	5.755605	0.883	0.3772	983.341
NZD_USD_return	-0.119754	0.403119	-0.297	0.7665	4.581
EUR_GBP_return	9.333065	7.367350	1.267	0.2054	672.208
EUR_JPY_return	-4.967159	10.576738	-0.470	0.6387	2148.816
GBP_JPY_return	8.243703	9.186616	0.897	0.3697	2226.819
AUD_JPY_return	-4.291870	5.773270	-0.743	0.4574	1094.774
CHF_JPY_return	5.701673	12.952212	0.440	0.6598	2693.764
SP500_return	-0.061236	0.842396	-0.073	0.9421	80.324
DowJones_return	-0.173600	0.497288	-0.349	0.7271	27.167
NASDAQ_return	-0.189586	1.322118	-0.143	0.8860	279.929
Russell2000_return	-0.004307	0.253764	-0.017	0.9865	11.825
XAU_USD_return	-0.343311	0.201475	-1.704	0.0886	2.257
Nasdaq100_return	0.277138	1.119250	0.248	0.8045	213.336
DJOilGas_return	0.104849	0.094143	1.114	0.2656	2.771
DJGoldMining_return	-0.031396	0.081658	-0.384	0.7007	1.968
Bitcoin_return	0.043132	0.050304	0.857	0.3913	3.173
Ethereum_return	0.307570	0.044741	6.874	9.02×10^{-12}	4.056
EthereumClassic_return	0.046611	0.027048	1.723	0.0850	2.280
Tether_return	-0.294985	0.334229	-0.883	0.3776	1.062
Cardano_return	0.458300	0.029423	15.576	$< 2 \times 10^{-16}$	2.434
Dogecoin_return	0.014405	0.011179	1.289	0.1978	1.219
Model Statistics					
Residual standard error: 4.728 on 1537 degrees of freedom					
Multiple R-squared: 0.5033					
Adjusted R-squared: 0.4949					
F-statistic: 59.91 on 26 and 1537 DF					
p-value:$< 2.2 \times 10^{-16}$					

suggests multicollinearity, which may affect the interpretation of individual coefficients and their signs. The remaining financial indices and cryptocurrencies do not demonstrate meaningful predictive power in this context.

Table 4.16: Selected Variables for Submodels on XRP

Criterion	**Variables Selected**
BIC	Ethereum, Cardano
AIC	EUR_USD, CAD_USD, AUD_USD, EUR_GBP, GBP_JPY, AUD_JPY, XAU_USD, Ethereum, Ethereum Classic, Cardano
LASSO	CAD_USD, AUD_USD, EUR_GBP, EUR_JPY, AUD_JPY, DowJones, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, Cardano, Dogecoin

The selected variables for each submodel selection method are summarized in Table 4.16. The submodel selected via AIC performs the best with the lowest average prediction error and standard deviation, making it the most effective model for predicting XRP daily returns, as shown in Table 4.17. Shrinkage estimators based on these submodels perform slightly worse but remain competitive. The Full Model shows the highest prediction errors and variability, while the other models generally offer moderate improvements.

4.2.6 A Model for Cardano

The full model estimation for Cardano's daily returns is presented in Table 4.18. The model identifies Bitcoin, Ethereum, Ethereum Classic, XRP, and Dogecoin returns as

Table 4.17: Average and Standard Deviation of Prediction Errors for XRP Daily Return

Estimator	**PE Average**	**PE s.d**
Full Model	23.97188	8.747485
Submodel BIC	23.34302	8.602202
Shrinkage Estimator BIC	23.48892	8.588637
Positive Shrinkage Estimator BIC	23.48706	8.586605
Submodel AIC	23.25886	8.520096
Shrinkage Estimator AIC	23.54466	8.555128
Positive Shrinkage Estimator AIC	23.54373	8.556902
Submodel LASSO	23.84955	8.782123
Shrinkage Estimator LASSO	23.87956	8.763229
Positive Shrinkage Estimator LASSO	23.86592	8.763299
LASSO	23.89894	8.787332
ENET	23.86923	8.798826
ALASSO	23.89547	8.774710
SCAD	23.74147	8.785486
RIDGE	23.79435	9.070347

statistically significant predictors, underscoring strong interdependencies among these cryptocurrencies. The model achieves an R^2 of 64.52%, indicating that a substantial portion of the variability in Cardano's returns is explained by the included predictors. However, the presence of high VIF values for several predictors suggests multicollinearity, which may compromise the reliability of individual coefficient estimates and their interpretability.

The selected variables for each submodel selection method are summarized in Table 4.19. The models corresponding to Shrinkage Estimator BIC and Positive Shrinkage Estimator BIC demonstrate the best performance with the lowest average prediction error and standard deviation, as presented in Table 4.20. The Full Model and models like ALASSO and SCAD show higher prediction errors and greater variability, indicating less effective and stable performance in predicting the daily return of Cardano.

4.2.7 A Model for Dogecoin

The full model estimation for Dogecoin's daily returns is presented in Table 4.21. The model identifies Bitcoin, Ethereum Classic, and Cardano returns as statistically significant predictors, suggesting that Dogecoin's performance is influenced by movements in these major cryptocurrencies. The model achieves an R^2 of 18.02%, indicating modest explanatory power. Several predictors exhibit high VIF values, which suggest potential multicollinearity and may affect the reliability and interpretability of individual coefficient estimates. Overall, while the selected predictors contribute meaningfully to the model, the relatively low R^2 suggests that other unobserved factors may also play a substantial role in determining Dogecoin's daily returns.

As shown in the Table 4.23, the Positive Shrinkage Estimator BIC, Positive Shrinkage Estimator AIC, and Submodel BIC models demonstrate the best performance with the lowest average prediction errors. The Full Model and SCAD show higher errors and variability, indicating less effective performance in predicting the daily return of Dogecoin. Models like LASSO, ENET, and ALASSO have moderate performance with slightly higher errors and variability.

4.3 Prediction Models for Next One Day

To explore the potential of forecasting for future for cryptocurrency market, this section focuses on predicting the next day's returns for the seven cryptocurrencies discussed earlier. This approach thus must use lagged predictors, specifically the financial indicators and cryptocurrency returns from the previous day, to estimate any future performance. For each cryptocurrency, the response variable is the return on day t, while the predictors include lagged returns (i.e on day $t-1$) of 20 financial indices (forex pairs, stock indices, and commodity indices) and similar lagged returns of all seven

cryptocurrencies. The full model, for example, for predicting Bitcoin's next-day return is given by,

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \\ &+ \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} \\ &+ \beta_{15} X_{15} + \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} \\ &+ \beta_{22} X_{22} + \beta_{23} X_{23} + \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \beta_{27} X_{27} + \varepsilon \end{aligned} \tag{4.3}$$

Where

- Y: Daily return of Bitcoin on day t
- X_1: Daily return of EUR/USD exchange rate on day t-1
- X_2: Daily return of USD/JPY exchange rate on day t-1
- X_3: Daily return of GBP/USD exchange rate on day t-1
- X_4: Daily return of CHF/USD exchange rate on day t-1
- X_5: Daily return of CAD/USD exchange rate on day t-1
- X_6: Daily return of AUD/USD exchange rate on day t-1
- X_7: Daily return of NZD/USD exchange rate on day t-1
- X_8: Daily return of EUR/GBP exchange rate on day t-1
- X_9: Daily return of EUR/JPY exchange rate on day t-1
- X_{10}: Daily return of GBP/JPY exchange rate on day t-1
- X_{11}: Daily return of AUD/JPY exchange rate on day t-1

- X_{12}: Daily return of CHF/JPY exchange rate on day t-1
- X_{13}: Daily return of S&P 500 Index on day t-1
- X_{14}: Daily return of Dow Jones Index on day t-1
- X_{15}: Daily return of NASDAQ Index on day t-1
- X_{16}: Daily return of Russell 2000 Index on day t-1
- X_{17}: Daily return of Gold price in USD (XAU/USD) on day t-1
- X_{18}: Daily return of NASDAQ 100 Index on day t-1
- X_{19}: Daily return of Dow Jones Oil and Gas Index on day t-1
- X_{20}: Daily return of Dow Jones Gold Mining Index on day t-1
- X_{21}: Daily return of Bitcoin on day t-1
- X_{22}: Daily return of Ethereum on day t-1
- X_{23}: Daily return of Ethereum Classic on day t-1
- X_{24}: Daily return of Tether on day t-1
- X_{25}: Daily return of XRP on day t-1
- X_{26}: Daily return of Cardano on day t-1
- X_{27}: Daily return of Dogecoin on day t-1
- $\beta_0, \ldots, \beta_{27}$: Unknown regression parameters
- ε: Random error term

The error term ε is assumed to be independently and normally distributed with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{1563\times 1} = \mathbf{X}_{1563\times 28}\beta_{28\times 1} + \varepsilon_{1563\times 1}, \;\; Rank(\mathbf{X}) = 28 < 1563 \tag{4.4}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin returns on day t, $\mathbf{X}$ is the design matrix of predictors containing 27 predictors (lagged returns on day $t-1$) and the intercept, β is the vector of unknown parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

The modeling framework remains consistent with earlier sections, incorporating full model estimation, submodel selection via AIC, BIC, and LASSO, and penalized regression techniques such as Ridge, LASSO, ENET, ALASSO, and SCAD. Post-shrinkage estimators are also employed to enhance robustness and predictive accuracy. As earlier, the model performance will evaluated using a bootstrap procedure with 250 iterations, allowing for a rigorous comparison of prediction errors across different estimation strategies.

We find that despite numerous variables in the model, none of the model has any respectable value of R^2. One may think that it may be due to large degrees of freedom assigned to error as the number of observations is very large. However, Adjusted R^2 values, which in some way account for large number of observations and number of variables also are quite low. Thus we conclude that none of the above models are suitable for one day forward prediction of daily returns. We thus refrain from discussing each model separately but present the corresponding model information from various analyses in Table 4.24 - Table 4.44.

That prediction is quite poor is also evident from the high average prediction errors reported in these tables. The takeaway from above analyses is that while movements of most cryptocurrencies seem to have co-movement with many other cryptocurrencies and other market indexes and variables, these co-movements do not extend to next day. Thus it is perhaps not possible to predict the corresponding returns for the immediate future (of one day forward). That brings the question: if not daily, can one make prediction of returns on weekly basis? This is the question we will address in our next chapter but for now, we will reanalyze the same data as above, by eliminating the problem of multicollinearity via principal component regression. It allows us to examine if we have not missed any signals due to multicollinearity and if the above conclusions remain the same.

Table 4.18: Coefficients and Model Statistics for the Full Model of Cardano

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.060053	0.098155	-0.612	0.5407	-
EUR_USD_return	11.710914	6.849745	1.710	0.0875	1044.096
USD_JPY_return	2.271410	9.447023	0.240	0.8100	2720.158
GBP_USD_return	-6.132115	5.012756	-1.223	0.2214	897.758
CHF_USD_return	-4.887371	10.419078	-0.469	0.6391	2422.036
CAD_USD_return	0.081499	0.357511	0.228	0.8197	2.616
AUD_USD_return	2.119661	4.637873	0.457	0.6477	983.706
NZD_USD_return	-0.276298	0.324706	-0.851	0.3949	4.579
EUR_GBP_return	-2.587181	5.938247	-0.436	0.6631	672.827
EUR_JPY_return	-9.266780	8.518497	-1.088	0.2768	2147.471
GBP_JPY_return	3.729089	7.402538	0.504	0.6145	2227.618
AUD_JPY_return	-1.981860	4.651805	-0.426	0.6701	1095.039
CHF_JPY_return	5.373256	10.434726	0.515	0.6067	2693.639
SP500_return	-0.006954	0.678678	-0.010	0.9918	80.324
DowJones_return	-0.164090	0.400635	-0.410	0.6822	27.166
NASDAQ_return	-0.598428	1.065064	-0.562	0.5743	279.875
Russell2000_return	0.284110	0.204317	1.391	0.1646	11.810
XAU_USD_return	-0.044416	0.162468	-0.273	0.7846	2.261
Nasdaq100_return	0.596234	0.901615	0.661	0.5085	213.284
DJOilGas_return	0.021701	0.075875	0.286	0.7749	2.773
DJGoldMining_return	0.015091	0.065790	0.229	0.8186	1.968
Bitcoin_return	0.315110	0.039733	7.931	4.16×10^{-15}	3.049
Ethereum_return	0.296086	0.035808	8.269	2.90×10^{-16}	4.002
EthereumClassic_return	0.110586	0.021629	5.113	3.57×10^{-7}	2.247
Tether_return	0.042743	0.269338	0.159	0.8739	1.062
XRP_return	0.297471	0.019098	15.576	$< 2 \times 10^{-16}$	1.739
Dogecoin_return	0.021432	0.008995	2.383	0.0173	1.215
Model Statistics					
Residual standard error: 3.809 on 1537 degrees of freedom					
Multiple R-squared: 0.6452					
Adjusted R-squared: 0.6392					
F-statistic: 107.5 on 26 and 1537 DF					
p-value: $< 2.2 \times 10^{-16}$					

Table 4.19: Selected Variables for Submodels on Cardano

Criterion	Variables Selected
BIC	Russell2000, Bitcoin, Ethereum, Ethereum Classic, XRP
AIC	EUR_USD, GBP_USD, EUR_JPY, GBP_JPY, Russell2000, Bitcoin, Ethereum, Ethereum Classic, XRP, Dogecoin
LASSO	CHF_USD, GBP_JPY, DowJones, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, XRP, Dogecoin

Table 4.20: Average and Standard Deviation of Prediction Errors for Cardano Daily Return

Estimator	PE Average	PE s.d
Full Model	14.45216	2.580331
Submodel BIC	14.29353	2.554071
Shrinkage Estimator BIC	14.26305	2.555757
Positive Shrinkage Estimator BIC	14.26146	2.555072
Submodel AIC	14.30014	2.575207
Shrinkage Estimator AIC	14.30718	2.579552
Positive Shrinkage Estimator AIC	14.30295	2.577682
Submodel LASSO	14.33792	2.576334
Shrinkage Estimator LASSO	14.34229	2.579207
Positive Shrinkage Estimator LASSO	14.33822	2.579562
LASSO	14.39627	2.590396
ENET	14.37920	2.583752
ALASSO	14.41187	2.591840
SCAD	14.42595	2.619588
RIDGE	14.35255	2.598300

Table 4.21: Coefficients and Model Statistics for the Full Model of Dogecoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.42637	0.27765	1.536	0.1248	
EUR_USD_return	9.20633	19.40545	0.474	0.6353	1045.928
USD_JPY_return	-20.68074	26.73543	-0.774	0.4393	2719.201
GBP_USD_return	-9.26335	14.19372	-0.653	0.5141	898.383
CHF_USD_return	-13.33261	29.49172	-0.452	0.6513	2422.060
CAD_USD_return	-1.21848	1.01149	-1.205	0.2285	2.613
AUD_USD_return	-7.26860	13.12724	-0.554	0.5799	983.644
NZD_USD_return	0.30048	0.91928	0.327	0.7438	4.581
EUR_GBP_return	4.41198	16.80908	0.262	0.7930	672.880
EUR_JPY_return	-14.16685	24.11848	-0.587	0.5570	2148.642
GBP_JPY_return	14.23114	20.95173	0.679	0.4971	2227.317
AUD_JPY_return	7.39995	13.16652	0.562	0.5742	1094.943
CHF_JPY_return	13.42839	29.53642	0.455	0.6494	2693.742
SP500_return	0.35965	1.92100	0.187	0.8515	80.322
DowJones_return	0.45784	1.13401	0.404	0.6865	27.167
NASDAQ_return	-1.18113	3.01486	-0.392	0.6953	279.905
Russell2000_return	0.05288	0.57869	0.091	0.9272	11.825
XAU_USD_return	-0.34166	0.45980	-0.743	0.4576	2.260
Nasdaq100_return	0.58047	2.55237	0.227	0.8201	213.337
DJOilGas_return	-0.03345	0.21477	-0.156	0.8762	2.773
DJGoldMining_return	0.21412	0.18614	1.150	0.2502	1.966
Bitcoin_return	0.62858	0.11362	5.532	3.71×10^{-8}	3.112
Ethereum_return	-0.19405	0.10347	-1.875	0.0609	4.171
EthereumClassic_return	0.37367	0.06100	6.126	1.15×10^{-9}	2.230
Tether_return	0.79636	0.76211	1.045	0.2962	1.062
XRP_return	0.07491	0.05814	1.289	0.1978	2.011
Cardano_return	0.17171	0.07207	2.383	0.0173	2.808

Model Statistics

Residual standard error: 10.78 on 1537 degrees of freedom
Multiple R-squared: 0.1802
Adjusted R-squared: 0.1664
F-statistic: 13 on 26 and 1537 DF
p-value: $< 2.2 \times 10^{-16}$

Table 4.22: Selected Variables for Submodels on Dogecoin

Criterion	Variables Selected
BIC	Bitcoin, Ethereum Classic, Cardano
AIC	Bitcoin, Ethereum, Ethereum Classic, Cardano
LASSO	CAD_USD, NZD_USD, EUR_GBP, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano

Table 4.23: Average and Standard Deviation of Prediction Errors for Dogecoin Daily Return

Estimator	PE Average	PE s.d
Full Model	122.8599	153.7917
Submodel BIC	122.0375	156.4631
Shrinkage Estimator BIC	122.1687	156.1581
Positive Shrinkage Estimator BIC	121.9551	155.9200
Submodel AIC	122.0520	155.5917
Shrinkage Estimator AIC	122.3725	155.9529
Positive Shrinkage Estimator AIC	122.0188	155.5387
Submodel LASSO	122.6352	154.3354
Shrinkage Estimator LASSO	122.7023	154.1834
Positive Shrinkage Estimator LASSO	122.6633	154.1399
LASSO	122.6766	154.1235
ENET	122.6073	154.2081
ALASSO	122.7131	154.1118
SCAD	122.8420	156.4566
RIDGE	122.1492	155.1279

Table 4.24: Coefficients and Model Statistics for the Full Model of Bitcoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.197092	0.108823	1.811	0.0703	-
EUR_USD_return	-7.578512	7.599570	-0.997	0.3188	1045.821
USD_JPY_return	-15.972649	10.471250	-1.525	0.1274	2720.116
GBP_USD_return	-7.317524	5.558971	-1.316	0.1883	898.664
CHF_USD_return	-0.631053	11.549098	-0.055	0.9564	2422.338
CAD_USD_return	0.596482	0.396264	1.505	0.1325	2.616
AUD_USD_return	-0.241249	5.141102	-0.047	0.9626	983.931
NZD_USD_return	-0.790794	0.360060	-2.196	0.0282	4.583
EUR_GBP_return	-4.506307	6.590220	-0.684	0.4942	674.457
EUR_JPY_return	12.671868	9.448686	1.341	0.1801	2149.297
GBP_JPY_return	2.578951	8.210068	0.314	0.7535	2229.981
AUD_JPY_return	-0.137618	5.156506	-0.027	0.9787	1095.226
CHF_JPY_return	0.721708	11.566613	0.062	0.9503	2693.526
SP500_return	0.119504	0.752317	0.159	0.8738	80.326
DowJones_return	-0.060930	0.444194	-0.137	0.8909	27.181
NASDAQ_return	-0.165100	1.182907	-0.140	0.8890	280.961
Russell2000_return	0.180294	0.226878	0.795	0.4269	11.852
XAU_USD_return	0.356024	0.180118	1.977	0.0483	2.261
Nasdaq100_return	0.005258	1.001371	0.005	0.9958	214.082
DJOilGas_return	-0.040497	0.084101	-0.482	0.6302	2.773
DJGoldMining_return	-0.064849	0.072927	-0.889	0.3740	1.968
Bitcoin_return	0.010018	0.044933	0.223	0.8236	3.174
Ethereum_return	-0.034283	0.040565	-0.845	0.3982	4.181
EthereumClassic_return	-0.018215	0.024184	-0.753	0.4515	2.286
Tether_return	0.636155	0.298530	2.131	0.0333	1.062
XRP_return	-0.001258	0.022793	-0.055	0.9560	2.015
Cardano_return	0.026378	0.028285	0.933	0.3512	2.821
Dogecoin_return	-0.002174	0.009989	-0.218	0.8277	1.220
Model Statistics					
Residual standard error: 4.222 on 1535 degrees of freedom					
Multiple R-squared: 0.02367					
Adjusted R-squared: 0.006499					
F-statistic: 1.378 on 27 and 1535 DF					
p-value: 0.09374					

Table 4.25: Selected Variables for Submodels on Bitcoin

Criterion	Variables Selected
BIC	NZD_USD
AIC	USD_JPY, GBP_USD, NZD_USD, EUR_GBP, EUR_JPY, XAU_USD, Tether
LASSO	EUR_USD, USD_JPY, GBP_USD, CHF_USD, CAD_USD, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.26: Average and Standard Deviation of Prediction Errors for Bitcoin Next One Day

Estimator	PE Average	PE s.d
Full Model	17.94247	2.71248
Submodel BIC	17.68691	2.73414
Shrinkage Estimator BIC	17.64807	2.69858
Positive Shrinkage Estimator BIC	17.64717	2.69722
Submodel AIC	17.59025	2.70036
Shrinkage Estimator AIC	17.63963	2.68236
Positive Shrinkage Estimator AIC	17.63710	2.68231
Submodel LASSO	17.91685	2.70564
Shrinkage Estimator LASSO	18.00889	2.71362
Positive Shrinkage Estimator LASSO	18.00889	2.71362
LASSO	17.90644	2.70946
ENET	17.89259	2.70986
ALASSO	17.87584	2.70290
SCAD	17.76646	2.74752
RIDGE	17.85795	2.71249

Table 4.27: Coefficients and Model Statistics for the Full Model of Ethereum

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.221400	0.137775	1.607	0.108268	-
EUR_USD_return	-10.547303	9.621427	-1.096	0.273150	1045.821
USD_JPY_return	-25.839701	13.257114	-1.949	0.051463	2720.116
GBP_USD_return	-10.567271	7.037929	-1.501	0.133439	898.664
CHF_USD_return	-11.099453	14.621722	-0.759	0.447905	2422.338
CAD_USD_return	0.644065	0.501689	1.284	0.199408	2.616
AUD_USD_return	6.741717	6.508887	1.036	0.300472	983.931
NZD_USD_return	-1.173022	0.455854	-2.573	0.010168	4.583
EUR_GBP_return	-1.151488	8.343541	-0.138	0.890251	674.457
EUR_JPY_return	12.271964	11.962499	1.026	0.305115	2149.297
GBP_JPY_return	9.228655	10.394348	0.888	0.374759	2229.981
AUD_JPY_return	-6.970258	6.528388	-1.068	0.285831	1095.226
CHF_JPY_return	11.137203	14.643897	0.761	0.447051	2693.526
SP500_return	1.121793	0.952470	1.178	0.239070	80.326
DowJones_return	-0.505740	0.562372	-0.899	0.368635	27.181
NASDAQ_return	-1.177028	1.497618	-0.786	0.432028	280.961
Russell2000_return	0.279688	0.287238	0.974	0.330351	11.852
XAU_USD_return	0.328003	0.228038	1.438	0.150534	2.261
Nasdaq100_return	0.485577	1.267785	0.383	0.701764	214.082
DJOilGas_return	-0.122188	0.106477	-1.148	0.251331	2.773
DJGoldMining_return	-0.033715	0.092329	-0.365	0.715043	1.968
Bitcoin_return	-0.023611	0.056887	-0.415	0.678156	3.174
Ethereum_return	0.021082	0.051358	0.410	0.681498	4.181
EthereumClassic_return	-0.046382	0.030618	-1.515	0.130014	2.286
Tether_return	1.324323	0.377954	3.504	0.000472	1.062
XRP_return	-0.010257	0.028857	-0.355	0.722295	2.015
Cardano_return	0.008332	0.035810	0.233	0.816038	2.821
Dogecoin_return	0.009320	0.012646	0.737	0.461257	1.220
Model Statistics					
Residual standard error: 5.345 on 1535 degrees of freedom					
Multiple R-squared: 0.03007					
Adjusted R-squared: 0.01301					
F-statistic: 1.762 on 27 and 1535 DF					
p-value: 0.009317					

Table 4.28: Selected Variables for Submodels on Ethereum

Criterion	Variables Selected
BIC	NZD_USD, Tether
AIC	USD_JPY, GBP_USD, NZD_USD, EUR_JPY, GBP_JPY, XAU_USD, Ethereum Classic, Tether
LASSO	EUR_USD, USD_JPY, GBP_USD, CAD_USD, AUD_USD, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.29: Average and Standard Deviation of Prediction Errors for Ethereum Next One Day

Estimator	PE Average	PE s.d
Full Model	28.99739	4.19828
Submodel BIC	28.52497	4.14348
Shrinkage Estimator BIC	28.50084	4.15266
Positive Shrinkage Estimator BIC	28.50066	4.15260
Submodel AIC	28.43181	4.14215
Shrinkage Estimator AIC	28.52516	4.14333
Positive Shrinkage Estimator AIC	28.51863	4.14304
Submodel LASSO	28.98726	4.19509
Shrinkage Estimator LASSO	48.22865	129.02434
Positive Shrinkage Estimator LASSO	48.22865	129.02434
LASSO	28.96522	4.19762
ENET	28.94239	4.19820
ALASSO	28.92572	4.19054
SCAD	28.75534	4.23617
RIDGE	28.92417	4.21958

Table 4.30: Coefficients and Model Statistics for the Full Model of Ethereum Classic

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.2051	0.1707	1.201	0.229812	
EUR_USD_return	-19.63	11.92	-1.647	0.099760	1045.821
USD_JPY_return	-27.06	16.43	-1.648	0.099609	2720.116
GBP_USD_return	-16.47	8.720	-1.889	0.059107	898.664
CHF_USD_return	-2.158	18.12	-0.119	0.905179	2422.338
CAD_USD_return	0.2919	0.6216	0.470	0.638721	2.616
AUD_USD_return	12.04	8.064	1.493	0.135673	983.931
NZD_USD_return	-1.073	0.5648	-1.900	0.057643	4.583
EUR_GBP_return	-3.846	10.34	-0.372	0.709908	674.457
EUR_JPY_return	23.59	14.82	1.592	0.111693	2149.297
GBP_JPY_return	12.44	12.88	0.966	0.334327	2229.981
AUD_JPY_return	-12.08	8.089	-1.494	0.135387	1095.226
CHF_JPY_return	2.634	18.14	0.145	0.884599	2693.526
SP500_return	0.6351	1.180	0.538	0.590518	80.326
DowJones_return	-0.09668	0.6968	-0.139	0.889666	27.181
NASDAQ_return	-2.718	1.856	-1.465	0.143152	280.961
Russell2000_return	0.4816	0.3559	1.353	0.176142	11.852
XAU_USD_return	-0.03567	0.2825	-0.126	0.899551	2.261
Nasdaq100_return	1.980	1.571	1.261	0.207615	214.082
DJOilGas_return	-0.04454	0.1319	-0.338	0.735721	2.773
DJGoldMining_return	-0.00004609	0.1144	0.000	0.999679	1.968
Bitcoin_return	-0.2141	0.07048	-3.038	0.002426	3.174
Ethereum_return	0.008774	0.06363	0.138	0.890351	4.181
EthereumClassic_return	0.1366	0.03794	3.602	0.000326	2.286
Tether_return	1.206	0.4683	2.576	0.010078	1.062
XRP_return	0.0005477	0.03575	0.015	0.987779	2.015
Cardano_return	-0.01552	0.04437	-0.350	0.726552	2.821
Dogecoin_return	0.02049	0.01567	1.308	0.191046	1.220
Model Statistics					
Residual standard error: 6.622 on 1535 degrees of freedom					
Multiple R-squared: 0.0338					
Adjusted R-squared: 0.0168					
F-statistic: 1.989 on 27 and 1535 DF					
p-value: 0.0019					

Table 4.31: Selected Variables for Submodels on Ethereum Classic

Criterion	Variables Selected
BIC	Bitcoin, Ethereum Classic
AIC	EUR_USD, USD_JPY, GBP_USD, AUD_USD, NZD_USD, EUR_JPY, GBP_JPY, AUD_JPY, NASDAQ, Russell2000, Nasdaq100, Bitcoin, Ethereum Classic, Tether
LASSO	EUR_USD, USD_JPY, GBP_USD, CHF_USD, CAD_USD, AUD_USD, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.32: Average and Standard Deviation of Prediction Errors for Ethereum Classic Next One Day

Estimator	PE Average	PE s.d
Full Model	44.94987	8.583617
Submodel BIC	44.10765	8.639135
Shrinkage Estimator BIC	44.11021	8.513619
Positive Shrinkage Estimator BIC	44.11021	8.513619
Submodel AIC	43.90129	8.485445
Shrinkage Estimator AIC	44.26009	8.473288
Positive Shrinkage Estimator AIC	44.25262	8.472928
Submodel LASSO	44.94987	8.583617
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	44.90795	8.588076
ENET	44.88445	8.590683
ALASSO	44.86185	8.578458
SCAD	44.72593	8.792237
RIDGE	44.95858	8.620026

Table 4.33: Coefficients and Model Statistics for the Full Model of Tether

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.0003	0.008843	0.034	0.9729	
EUR_USD_return	0.2524	0.6175	0.409	0.6828	1045.821
USD_JPY_return	0.01625	0.8509	0.019	0.9848	2720.116
GBP_USD_return	-0.5312	0.4517	-1.176	0.2398	898.664
CHF_USD_return	-0.3298	0.9384	-0.351	0.7253	2422.338
CAD_USD_return	-0.001861	0.03220	-0.058	0.9539	2.616
AUD_USD_return	0.5850	0.4178	1.400	0.1616	983.931
NZD_USD_return	0.04150	0.02926	1.418	0.1563	4.583
EUR_GBP_return	0.1300	0.5355	0.243	0.8082	674.457
EUR_JPY_return	-0.4611	0.7678	-0.601	0.5482	2149.297
GBP_JPY_return	0.6490	0.6671	0.973	0.3308	2229.981
AUD_JPY_return	-0.5542	0.4190	-1.323	0.1862	1095.226
CHF_JPY_return	0.3314	0.9399	0.353	0.7245	2693.526
SP500_return	0.04646	0.06113	0.760	0.4474	80.326
DowJones_return	-0.02024	0.03609	-0.561	0.5750	27.181
NASDAQ_return	0.004805	0.09612	0.050	0.9601	280.961
Russell2000_return	0.001057	0.01844	0.057	0.9543	11.852
XAU_USD_return	0.01276	0.01464	0.872	0.3834	2.261
Nasdaq100_return	-0.02070	0.08137	-0.254	0.7992	214.082
DJOilGas_return	-0.01083	0.006834	-1.585	0.1132	2.773
DJGoldMining_return	-0.01219	0.005926	-2.057	0.0398	1.968
Bitcoin_return	0.0003353	0.003651	0.092	0.9268	3.174
Ethereum_return	0.00003293	0.003296	0.010	0.9920	4.181
EthereumClassic_return	-0.001492	0.001965	-0.759	0.4479	2.286
Tether_return	-0.3566	0.02426	-14.701	$< 2 \times 10^{-16}$	1.062
XRP_return	-0.002128	0.001852	-1.149	0.2508	2.015
Cardano_return	-0.00009587	0.002298	-0.042	0.9667	2.821
Dogecoin_return	0.0008553	0.0008116	1.054	0.2921	1.220
Model Statistics					
Residual standard error: 0.3431 on 1535 degrees of freedom					
Multiple R-squared: 0.1469					
Adjusted R-squared: 0.1319					
F-statistic: 9.79 on 27 and 1535 DF					
p-value: $< 2.2 \times 10^{-16}$					

Table 4.34: Selected Variables for Submodels on Tether

Criterion	Variables Selected
BIC	Tether
AIC	GBP_USD, AUD_USD, NZD_USD, EUR_JPY, GBP_JPY, AUD_JPY, DJOilGas, DJGoldMining, Tether, XRP
LASSO	No predictors selected

Table 4.35: Average and Standard Deviation of Prediction Errors for Tether Next One Day

Estimator	PE Average	PE s.d
Full Model	0.1196693	0.04405945
Submodel BIC	0.1154040	0.04400552
Shrinkage Estimator BIC	0.1168941	0.04394123
Positive Shrinkage Estimator BIC	0.1168941	0.04394123
Submodel AIC	0.1158488	0.04270005
Shrinkage Estimator AIC	0.1173113	0.04359966
Positive Shrinkage Estimator AIC	0.1173037	0.04359902
Submodel LASSO	-	-
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	0.1189798	0.04768381
ENET	0.1188354	0.04786093
ALASSO	0.1187297	0.04682008
SCAD	0.1171025	0.04416081
RIDGE	0.1179344	0.04408067

Table 4.36: Coefficients and Model Statistics for the Full Model of XRP

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.188547	0.170736	1.104	0.2696	
EUR_USD_return	-11.477180	11.923228	-0.963	0.3359	1045.821
USD_JPY_return	-8.358778	16.428706	-0.509	0.6110	2720.116
GBP_USD_return	-6.860998	8.721662	-0.787	0.4316	898.664
CHF_USD_return	14.140761	18.119779	0.780	0.4353	2422.338
CAD_USD_return	0.338597	0.621712	0.545	0.5861	2.616
AUD_USD_return	-3.720125	8.066053	-0.461	0.6447	983.931
NZD_USD_return	-0.531995	0.564911	-0.942	0.3465	4.583
EUR_GBP_return	-8.999603	10.339624	-0.870	0.3842	674.457
EUR_JPY_return	21.505959	14.824371	1.451	0.1471	2149.297
GBP_JPY_return	-2.349068	12.881060	-0.182	0.8553	2229.981
AUD_JPY_return	3.279105	8.090220	0.405	0.6853	1095.226
CHF_JPY_return	-14.476773	18.147259	-0.798	0.4251	2693.526
SP500_return	0.322758	1.180336	0.273	0.7845	80.326
DowJones_return	-0.186252	0.696912	-0.267	0.7893	27.181
NASDAQ_return	0.315642	1.855904	0.170	0.8650	280.961
Russell2000_return	0.281630	0.355956	0.791	0.4290	11.852
XAU_USD_return	0.061121	0.282593	0.216	0.8288	2.261
Nasdaq100_return	-0.710525	1.571086	-0.452	0.6512	214.082
DJOilGas_return	0.012329	0.131950	0.093	0.9256	2.773
DJGoldMining_return	0.090854	0.114418	0.794	0.4273	1.968
Bitcoin_return	-0.002968	0.070496	-0.042	0.9664	3.174
Ethereum_return	-0.036347	0.063644	-0.571	0.5680	4.181
EthereumClassic_return	-0.082806	0.037943	-2.182	0.0292	2.286
Tether_return	0.779026	0.468375	1.663	0.0965	1.062
XRP_return	0.019363	0.035761	0.541	0.5883	2.015
Cardano_return	0.053936	0.044377	1.215	0.2244	2.821
Dogecoin_return	0.009975	0.015671	0.637	0.5245	1.220
Model Statistics					
Residual standard error: 6.624 on 1535 degrees of freedom					
Multiple R-squared: 0.01751					
Adjusted R-squared: 0.0002281					
F-statistic: 1.013 on 27 and 1535 DF					
p-value: 0.4458					

Table 4.37: Selected Variables for Submodels on XRP

Criterion	Variables Selected
BIC	No predictors selected
AIC	Russell2000, Nasdaq100, Ethereum Classic, Tether, Cardano, AUD_JPY
LASSO	EUR_USD, USD_JPY, GBP_USD, CHF_USD, CAD_USD, AUD_USD, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.38: Average and Standard Deviation of Prediction Errors for XRP Next One Day

Estimator	PE Average	PE s.d
Full Model	46.07904	14.07919
Submodel BIC	-	-
Shrinkage Estimator BIC	-	-
Positive Shrinkage Estimator BIC	-	-
Submodel AIC	86.89743	49.48840
Shrinkage Estimator AIC	63.18758	31.44557
Positive Shrinkage Estimator AIC	61.83743	28.74659
Submodel LASSO	46.07904	14.07919
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	46.02797	14.06553
ENET	46.00097	14.06275
ALASSO	45.97195	14.06066
SCAD	45.42932	14.02400
RIDGE	45.90600	14.04316

Table 4.39: Coefficients and Model Statistics for the Full Model of Cardano

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.198489	0.160709	1.235	0.21699	
EUR_USD_return	-29.443210	11.223017	-2.623	0.00879	1045.821
USD_JPY_return	-25.651507	15.463904	-1.659	0.09736	2720.116
GBP_USD_return	-4.795754	8.209468	-0.584	0.55919	898.664
CHF_USD_return	3.568475	17.055666	0.209	0.83430	2422.338
CAD_USD_return	0.632910	0.585201	1.082	0.27963	2.616
AUD_USD_return	5.707712	7.592361	0.752	0.45230	983.931
NZD_USD_return	-1.163564	0.531736	-2.188	0.02880	4.583
EUR_GBP_return	-1.906280	9.732413	-0.196	0.84474	674.457
EUR_JPY_return	32.647637	13.953785	2.340	0.01943	2149.297
GBP_JPY_return	2.654558	12.124599	0.219	0.82673	2229.981
AUD_JPY_return	-5.518444	7.615109	-0.725	0.46876	1095.226
CHF_JPY_return	-4.343760	17.081531	-0.254	0.79930	2693.526
SP500_return	0.480819	1.111019	0.433	0.66524	80.326
DowJones_return	-0.088678	0.655984	-0.135	0.89248	27.181
NASDAQ_return	0.288590	1.746913	0.165	0.86881	280.961
Russell2000_return	0.072079	0.335052	0.215	0.82970	11.852
XAU_USD_return	0.180705	0.265998	0.679	0.49702	2.261
Nasdaq100_return	-0.676964	1.478821	-0.458	0.64718	214.082
DJOilGas_return	-0.077205	0.124201	-0.622	0.53429	2.773
DJGoldMining_return	-0.027037	0.107699	-0.251	0.80181	1.968
Bitcoin_return	-0.088541	0.066356	-1.334	0.18229	3.174
Ethereum_return	-0.044217	0.059907	-0.738	0.46056	4.181
EthereumClassic_return	0.002579	0.035715	0.072	0.94245	2.286
Tether_return	1.497362	0.440869	3.396	0.00070	1.062
XRP_return	-0.006589	0.033661	-0.196	0.84483	2.015
Cardano_return	0.059920	0.041771	1.434	0.15164	2.821
Dogecoin_return	-0.004803	0.014751	-0.326	0.74477	1.220
Model Statistics					
Residual standard error: 6.235 on 1535 degrees of freedom					
Multiple R-squared: 0.02849					
Adjusted R-squared: 0.0114					
F-statistic: 1.667 on 27 and 1535 DF					
p-value: 0.01737					

Table 4.40: Selected Variables for Submodels on Cardano

Criterion	Variables Selected
BIC	EUR_USD, USD_JPY, NZD_USD, EUR_JPY, Tether
AIC	EUR_USD, USD_JPY, NZD_USD, EUR_JPY, Bitcoin, Tether
LASSO	EUR_USD, USD_JPY, GBP_USD, CAD_USD, AUD_USD, NZD_USD, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.41: Average and Standard Deviation of Prediction Errors for Cardano Next One Day

Estimator	PE Average	PE s.d
Full Model	39.24932	4.854843
Submodel BIC	38.62508	4.743032
Shrinkage Estimator BIC	38.62550	4.724095
Positive Shrinkage Estimator BIC	38.61815	4.730238
Submodel AIC	38.56733	4.768260
Shrinkage Estimator AIC	38.60655	4.743819
Positive Shrinkage Estimator AIC	38.59783	4.749374
Submodel LASSO	39.20057	4.859697
Shrinkage Estimator LASSO	39.60958	5.515323
Positive Shrinkage Estimator LASSO	39.60958	5.515323
LASSO	39.20883	4.861533
ENET	39.18443	4.858242
ALASSO	39.15330	4.860119
SCAD	39.14531	4.846427
RIDGE	39.21972	4.875744

Table 4.42: Coefficients and Model Statistics for the Full Model of Dogecoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.56387	0.30270	1.863	0.0627	
EUR_USD_return	-39.39825	21.13901	-1.864	0.0625	1045.821
USD_JPY_return	-40.05551	29.12689	-1.375	0.1693	2720.116
GBP_USD_return	14.63192	15.46287	0.946	0.3442	898.664
CHF_USD_return	-24.15663	32.12504	-0.752	0.4522	2422.338
CAD_USD_return	-0.27315	1.10225	-0.248	0.8043	2.616
AUD_USD_return	8.85722	14.30052	0.619	0.5358	983.931
NZD_USD_return	-1.02620	1.00155	-1.025	0.3057	4.583
EUR_GBP_return	19.28658	18.33139	1.052	0.2929	674.457
EUR_JPY_return	19.72959	26.28252	0.751	0.4530	2149.297
GBP_JPY_return	5.60834	22.83717	0.246	0.8060	2229.981
AUD_JPY_return	-9.29540	14.34337	-0.648	0.5170	1095.226
CHF_JPY_return	24.34215	32.17376	0.757	0.4494	2693.526
SP500_return	-4.08680	2.09265	-1.953	0.0510	80.326
DowJones_return	2.03076	1.23557	1.644	0.1005	27.181
NASDAQ_return	-0.46634	3.29038	-0.142	0.8873	280.961
Russell2000_return	0.64896	0.63108	1.028	0.3040	11.852
XAU_USD_return	0.72080	0.50102	1.439	0.1504	2.261
Nasdaq100_return	1.66133	2.78542	0.596	0.5510	214.082
DJOilGas_return	0.07859	0.23394	0.336	0.7369	2.773
DJGoldMining_return	-0.22199	0.20285	-1.094	0.2740	1.968
Bitcoin_return	-0.23255	0.12498	-1.861	0.0630	3.174
Ethereum_return	-0.05395	0.11284	-0.478	0.6326	4.181
EthereumClassic_return	0.09665	0.06727	1.437	0.1510	2.286
Tether_return	0.38213	0.83039	0.460	0.6454	1.062
XRP_return	-0.09097	0.06340	-1.435	0.1515	2.015
Cardano_return	0.08279	0.07868	1.052	0.2928	2.821
Dogecoin_return	0.11124	0.02778	4.004	6.53×10^{-5}	1.220
Model Statistics					
Residual standard error: 11.74 on 1535 degrees of freedom					
Multiple R-squared: 0.02869					
Adjusted R-squared: 0.01161					
F-statistic: 1.68 on 27 and 1535 DF					
p-value: 0.01605					

Table 4.43: Selected Variables for Submodels on Dogecoin

Criterion	Variables Selected
BIC	Bitcoin, Dogecoin
AIC	EUR_USD, USD_JPY, NZD_USD, EUR_JPY, SP500, DowJones, Russell2000, Nasdaq100, Bitcoin, Dogecoin
LASSO	EUR_USD, USD_JPY, GBP_USD, CAD_USD, AUD_USD, NZD_USD, EUR_GBP, EUR_JPY, GBP_JPY, AUD_JPY, CHF_JPY, SP500, DowJones, NASDAQ, Russell2000, XAU_USD, Nasdaq100, DJOilGas, DJGoldMining, Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin

Table 4.44: Average and Standard Deviation of Prediction Errors for Dogecoin Next One Day

Estimator	PE Average	PE s.d
Full Model	150.3232	167.5707
Submodel BIC	149.3098	171.0377
Shrinkage Estimator BIC	148.8897	169.3190
Positive Shrinkage Estimator BIC	148.8896	169.3191
Submodel AIC	149.0622	168.3811
Shrinkage Estimator AIC	149.3100	168.4289
Positive Shrinkage Estimator AIC	149.2739	168.4077
Submodel LASSO	150.3270	167.5282
Shrinkage Estimator LASSO	439.1845	3606.9590
Positive Shrinkage Estimator LASSO	439.1845	3606.9590
LASSO	150.1274	167.5735
ENET	149.9838	167.6422
ALASSO	150.1085	167.6614
SCAD	150.1313	171.7698
RIDGE	149.6716	169.0193

4.4 Regression Models with Principal Components

To address dimensionality and multicollinearity challenges inherent in financial data, this section introduces a regression framework that incorporates principal components derived from major market indices as explanatory variables. We initiate the process by applying the Principal Component Analyses separately to three separate categories of predictors, namely the set of currency pairs, various stock indices, and various commodity indices (such as gold and oil), using their respective covariance matrices. From each Principal Component Analysis, the first principal component is extracted, capturing the major part of total variance within each group of explanatary variables. These principal components are themselves used as explanatory variables.

To model the returns of a given cryptocurrency, these principal components obtained as above are then integrated with the daily returns of six other major cryptocurrencies to construct a reduced predictor set. Linear regression models are subsequently fitted for each cryptocurrency. This approach mitigates the problems related to multicollinearity. Specifically, we have used covariance matrix for the three principal component analysis to preserve variance magnitude and uncover latent financial factors. The Table 4.45 gives the variables used.

And the model, for example, for Bitcoin is given by,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \varepsilon \quad (4.5)$$

Where

- Y: Daily return of Bitcoin
- X_1: Currency_PC1 value

Table 4.45: Principal Component and Cryptocurrency Predictors

Predictor	Description
Currency_PC1	First principal component derived from the covariance matrix of daily returns for the following currency pairs: EUR/USD, USD/JPY, GBP/USD, CHF/USD, CAD/USD, AUD/USD, NZD/USD, EUR/GBP, EUR/JPY, GBP/JPY, AUD/JPY, CHF/JPY. (Explains 47.54% of the total variability.)
Stock_PC1	First principal component derived from daily returns of various stock indices: S&P 500, Dow Jones, NASDAQ, Nasdaq 100, Russell 2000. (Explains 90.19% of the total variability.)
GoldOils_PC1	First principal component derived from daily returns of various commodity indices: DJ Oil & Gas, XAU/USD, DJ Gold Mining. (Explains 57.68% of the total variability.)
Daily returns from [*]: Bitcoin, Ethereum, Ethereum Classic, Tether, XRP, Cardano, Dogecoin.	

[*] One of the following is used as the response variable and other six as the predictors, thereby leading to seven separate analyses.

- X_2: Stock_PC1 value
- X_3: GoldOils_PC1 value
- X_4: Daily return of Ethereum
- X_5: Daily return of Ethereum Classic
- X_6: Daily return of Tether
- X_7: Daily return of XRP
- X_8: Daily return of Cardano
- X_9: Daily return of Dogecoin
- $\beta_0, \ldots, \beta_9$: Unknown regression parameters

- ε: Random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{1564\times1} = \mathbf{X}_{1564\times10}\beta_{10\times1} + \varepsilon_{1564\times1}, \;\; Rank(\mathbf{X}) = 10 < 1564 \tag{4.6}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin daily returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 10$ predictors including the intercept, β is the vector of unknown parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

Table 4.46: Importance of components in PCA of currency pairs

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**	**Comp.6**
Standard deviation	1.3031	0.9690	0.6073	0.5643	0.3258	0.2847
Proportion of Variance	0.4754	0.2629	0.1033	0.0891	0.0297	0.0227
Cumulative Proportion	0.4754	0.7382	0.8415	0.9306	0.9604	0.9831
	Comp.7	**Comp.8**	**Comp.9**	**Comp.10**	**Comp.11**	**Comp.12**
Standard deviation	0.2447	0.0155	0.0136	0.0102	0.0084	0.0059
Proportion of Variance	0.0168	0.0001	0.0001	0.0000	0.0000	0.0000
Cumulative Proportion	0.9998	0.9999	1.0000	1.0000	1.0000	1.0000

The PCA results for currency pairs, stock indices, and commodity indices are presented in Tables 4.46 - 4.48. For currency pairs, the first principal component (Comp.1) exhibits a standard deviation of 1.3031, accounting for approximately 47.54% of the total variance. The first principal component derived from stock indices exhibits a

notably high standard deviation of 3.0824, capturing about 90.19% of the total variance, indicating that a single component effectively summarizes the return behavior of the stock indices. For gold and oil indices, the first principal component has a standard deviation of 2.3384 and explains 57.68% of the variance. Following the extraction of the leading principal component from each group, these components are then used along with the daily returns of seven major cryptocurrencies, to construct a unified dataset. The linear regression model given in equation 4.7 for the daily returns of a given cryptocurrency, using the daily returns of the six other cryptocurrencies and the three principal components as predictors. This approach aims to enhance prediction performance by leveraging the reduced dimensionality and mitigated collinearity achieved through PCA, providing a more robust framework for modeling cryptocurrency return dynamics. As we shall see the multicollinearity has been drastically reduced in each case. Thus the coefficients of predictors can be interpreted more unambiguously.

Table 4.47: Importance of components in PCA of stock indices

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**
Standard deviation	3.0824	0.8372	0.5580	0.1279	0.0700
Proportion of Variance	0.9019	0.0665	0.0296	0.0016	0.0005
Cumulative Proportion	0.9019	0.9684	0.9980	0.9995	1.0000

4.4.1 Models for the Present-Day Returns

To evaluate the predictive relationships between cryptocurrency returns and broader financial market dynamics, we consider the regression models in which the

Table 4.48: Importance of components in PCA of Gold and Oil indices

	Comp.1	**Comp.2**	**Comp.3**
Standard deviation	2.3384	1.8994	0.6363
Proportion of Variance	0.5768	0.3805	0.0427
Cumulative Proportion	0.5768	0.9573	1.0000

present-day return of each of seven major cryptocurrencies served as the response variable. As indicated earlier, the explanatory variables comprised the present-day returns of the other six cryptocurrencies, along with three principal components extracted from currency pairs, stock indices, and commodity indices (gold and oil), respectively.

Table 4.49: Model Statistics for Bitcoin Model with Principal Components

Variable	**Estimate**	**Std. Error**	**t value**	**P value**	**VIF**
(Intercept)	0.0377	0.0607	0.621	0.5345	
Currency_PC1	0.0197	0.0583	0.337	0.7358	1.569
Stock_PC1	0.0313	0.0255	1.225	0.2208	1.687
GoldOils_PC1	0.0431	0.0323	1.335	0.1822	1.555
Ethereum_return	0.4587	0.0197	23.321	$< 2 \times 10^{-16}$	3.051
EthereumClassic_return	0.0261	0.0136	1.919	0.0552	2.239
Tether_return	0.2114	0.1663	1.272	0.2037	1.023
XRP_return	0.0109	0.0129	0.849	0.3963	1.992
Cardano_return	0.1263	0.0157	8.055	1.57×10^{-15}	2.689
Dogecoin_return	0.0310	0.0056	5.532	3.71×10^{-8}	1.191
Model Statistics					
Residual standard error: 2.396 on 1554 degrees of freedom					
Multiple R-squared: 0.6816					
Adjusted R-squared: 0.6797					
F-statistic: 369.5 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 6.0320, Standard deviation 1.1641					

The regression results reveal that inter-cryptocurrency relationships play dominant roles. Results of Bitcoin are presented in Table 4.49, multicollinearity has been substantially reduced. Ethereum, Cardano and Dogecoin exhibit statistically significant positive influence on Bitcoin's daily returns, with Ethereum exhibiting the strongest influence, suggesting their comovements. Other indices represented by principal components show limited predictive power. The model explains approximately 68.16% of Bitcoin's return total variability, compared to 68.50% reported by full model (Table 4.3). Thus there is very little loss in model fit. The bootstrap average prediction error is 6.0320, slightly lower than the 6.0555 from the full model, indicating marginally improved predictive performance. These results suggests that Bitcoin's short-term movements are more closely tied to internal market dynamics among cryptocurrencies rather than various external financial indicators.

As shown in Table 4.50, in case of Ethereum's daily returns are significantly influenced by several key factors. Bitcoin, Ethereum Classic, XRP, and Cardano returns exhibit strong positive relationships with Ethereum, with Bitcoin showing the most substantial impact, highlighting the interconnected nature of the cryptocurrency market. Additionally, the first principal component corresponding to various stock indices is a significant predictor, suggesting that macroeconomic factors may also affect the Ethereum's performance. While Dogecoin returns demonstrate a marginal negative impact, currency and gold/oil principal components, as well as Tether returns, do not show significance. The model explains approximately 75.72% of Ethereum's return variability, providing a comprehensive understanding of the factors driving its market dynamics. Quality of fit of the model is not much different than that for full model (See Table 4.6, $R^2 = 0.7608$). The bootstrap average prediction error is 7.3020, slightly higher than the 7.2506 from the full model, suggesting comparable predictive performance.

Table 4.50: Model Statistics for Ethereum Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.0346	0.0674	0.513	0.6080	
Currency_PC1	0.0570	0.0646	0.882	0.3781	1.568
Stock_PC1	0.0624	0.0283	2.205	0.0276	1.683
GoldOils_PC1	-0.0208	0.0359	-0.578	0.5630	1.556
Bitcoin_return	0.5652	0.0242	23.321	$< 2 \times 10^{-16}$	2.326
EthereumClassic_return	0.2213	0.0140	15.795	$< 2 \times 10^{-16}$	1.934
Tether_return	-0.1040	0.1846	-0.563	0.5734	1.024
XRP_return	0.0954	0.0141	6.777	1.73×10^{-11}	1.936
Cardano_return	0.1435	0.0174	8.257	3.15×10^{-16}	2.684
Dogecoin_return	-0.0123	0.0063	-1.958	0.0504	1.211
Model Statistics					
Residual standard error: 2.66 on 1554 degrees of freedom					
Multiple R-squared: 0.7572					
Adjusted R-squared: 0.7558					
F-statistic: 538.5 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 7.3020, Standard deviation 1.6512					

For Ethereum Classic, the regression analysis indicates that the Ethereum's returns have the most statistically significant positive influence, underscoring a strong correlation in their market behavior (see Table 4.51). Cardano and Dogecoin also demonstrate significant positive associations with Ethereum Classic, further highlighting the interconnected nature within the cryptocurrency market. Bitcoin's returns show a marginal effect, while XRP, though not statistically significant, approaches relevance. In contrast, the three first principal components derived from currency pairs, stock indices, and commodity indices as well as Tether's returns do not exhibit meaningful predictive power. Overall, the model accounts for approximately 55.44% of the variability in Ethereum Classic's returns, which is nearly identical to the 0.5623 reported for the full model, suggesting it captures a moderate portion of the return dynamics influenced

Table 4.51: Model Statistics for Ethereum Classic Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.0298	0.1132	-0.263	0.7925	
Currency_PC1	-0.0478	0.1086	-0.440	0.6597	1.569
Stock_PC1	-0.0124	0.0476	-0.260	0.7948	1.688
GoldOils_PC1	0.0613	0.0603	1.018	0.3090	1.555
Bitcoin_return	0.0907	0.0473	1.919	0.0552	3.133
Ethereum_return	0.6249	0.0396	15.795	$< 2 \times 10^{-16}$	3.549
Tether_return	-0.2001	0.3102	-0.645	0.5190	1.024
XRP_return	0.0399	0.0240	1.666	0.0959	1.989
Cardano_return	0.1544	0.0296	5.217	2.06×10^{-7}	2.753
Dogecoin_return	0.0641	0.0104	6.145	1.01×10^{-9}	1.186
Model Statistics					
Residual standard error: 4.47 on 1554 degrees of freedom					
Multiple R-squared: 0.5544					
Adjusted R-squared: 0.5518					
F-statistic: 214.8 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 20.6323, Standard deviation 5.8583					

primarily by specific cryptocurrencies rather than broader macroeconomic factors. The bootstrap average prediction error is 20.6323, slightly lower than the 20.8120 from the full model, suggesting a modest improvement in predictive accuracy.

The regression model results for Tether's daily returns is presented in Table 4.52. Notably, the model shows that the currency principal component has a positive and statistically significant effect on Tether's returns, while the stock principal component has a negative and significant impact. These findings suggest that Tether may be modestly responsive to broader macroeconomic signals, particularly those captured by currency and equity markets. In contrast, none of the cryptocurrency returns are significant predictors of Tether's returns, indicating that Tether's behavior may be less influenced by the fluctuations of other cryptocurrencies. With only 2.34% of the variability in returns

Table 4.52: Model Statistics for Tether Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.000048	0.009256	0.005	0.996	
Currency_PC1	0.01892	0.008869	2.134	0.033	1.565
Stock_PC1	-0.02086	0.003859	-5.405	7.47×10^{-8}	1.657
GoldOils_PC1	0.001913	0.004929	0.388	0.698	1.556
Bitcoin_return	0.004916	0.003866	1.272	0.204	3.137
Ethereum_return	-0.001962	0.003484	-0.563	0.573	4.118
EthereumClassic_return	-0.001337	0.002074	-0.645	0.519	2.244
XRP_return	-0.001734	0.001961	-0.884	0.377	1.992
Cardano_return	0.000678	0.002440	0.278	0.781	2.801
Dogecoin_return	0.000816	0.000862	0.946	0.344	1.214
Model Statistics					
Residual standard error: 0.3654 on 1554 degrees of freedom					
Multiple R-squared: 0.02343					
Adjusted R-squared: 0.01777					
F-statistic: 4.143 on 9 and 1554 DF					
p-value: 2.714×10^{-5}					
Bootstrap sample prediction errors: Average 0.1379, Standard deviation 0.0527					

explained, the model exhibits limited explanatory power. The bootstrap average prediction error is 0.1379, marginally higher than the 0.1358 from the full model, suggesting comparable predictive performance.

The regression analysis reveals that Ethereum and Cardano returns are statistically significant predictors of XRP's daily returns, underscoring strong interdependencies among these assets, as presented in The Table 4.53. Ethereum Classic also approaches significance, suggesting a potential secondary influence. Other variables, including principal components derived from currency pairs, stock indices, and commodity indices, do not exhibit meaningful predictive power. With approximately 49.82% of the variability in XRP's returns explained, the model highlights the dominant role of specific cryptocurrency relationships, while broader macroeconomic factors appear to have limited

Table 4.53: Model Statistics for XRP Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.004713	0.119724	0.039	0.9686	
Currency_PC1	0.075455	0.114870	0.657	0.5114	1.569
Stock_PC1	0.020758	0.050373	0.412	0.6803	1.688
GoldOils_PC1	-0.087723	0.063720	-1.377	0.1688	1.554
Bitcoin_return	0.042444	0.050019	0.849	0.3963	3.139
Ethereum_return	0.301043	0.044420	6.777	1.73×10^{-11}	4.000
EthereumClassic_return	0.044647	0.026800	1.666	0.0959	2.240
Tether_return	-0.290118	0.328021	-0.884	0.3766	1.023
Cardano_return	0.462256	0.029295	15.780	$< 2 \times 10^{-16}$	2.415
Dogecoin_return	0.014901	0.011150	1.336	0.1816	1.213
Model Statistics					
Residual standard error: 4.726 on 1554 degrees of freedom					
Multiple R-squared: 0.4982					
Adjusted R-squared: 0.4953					
F-statistic: 171.4 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 24.0253, Standard deviation 8.8190					

relevance in shaping XRP's short-term performance. Compared to the R^2 of 0.5033 reported for the full model, it suggests a negligible reduction in explanatory strength. The bootstrap average prediction error is 24.0253, marginally higher than the 23.9719 from the full model, indicating comparable predictive performance.

The Table 4.54 highlights several key influences on Cardano's returns. XRP, Bitcoin, Ethereum, and Ethereum Classic returns are statistically significant predictors of Cardano's daily returns, with Dogecoin also showing a positive but comparatively weaker influence. The stock principal component approaches significance, suggesting a potential link between Cardano's performance and broader equity market trends. The model explains approximately 64.3% of the variability in Cardano's returns, which is nearly identical to the 0.6452 reported for the full model, underscoring the dominant role of

Table 4.54: Model Statistics for Cardano Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.043828	0.096243	-0.455	0.6489	
Currency_PC1	0.093797	0.092329	1.016	0.3098	1.568
Stock_PC1	0.067742	0.040462	1.674	0.0943	1.685
GoldOils_PC1	0.019709	0.051255	0.385	0.7006	1.556
Bitcoin_return	0.317417	0.039406	8.055	1.57×10^{-15}	3.014
Ethereum_return	0.292841	0.035465	8.257	3.15×10^{-16}	3.946
EthereumClassic_return	0.111539	0.021378	5.217	2.06×10^{-7}	2.206
Tether_return	0.073313	0.263764	0.278	0.7811	1.024
XRP_return	0.298754	0.018933	15.780	$< 2 \times 10^{-16}$	1.718
Dogecoin_return	0.021610	0.008952	2.414	0.0159	1.210
Model Statistics					
Residual standard error: 3.8 on 1554 degrees of freedom					
Multiple R-squared: 0.643					
Adjusted R-squared: 0.641					
F-statistic: 311 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 14.9488, Standard deviation 2.7535					

inter-cryptocurrency relationships while highlighting the relatively limited impact of macroeconomic factors captured by principal components. The bootstrap average prediction error is 14.9488, slightly higher than the 14.4522 from the full model, suggesting comparable predictive performance.

Regression results, detailed in Table 4.55, show that Bitcoin and Ethereum Classic returns are significant predictors of Dogecoin's daily returns, with Bitcoin showing a strong positive influence and Ethereum Classic contributing substantially. Cardano also demonstrates a significant positive association, though to a lesser extent. While these relationships highlight meaningful intra-market dependencies, the model explains only about 17.6% of the variability in Dogecoin's returns. This limited explanatory power suggests that Dogecoin's performance is likely shaped by additional factors not captured

Table 4.55: Model Statistics for Dogecoin Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.416500	0.272020	1.531	0.1259	
Currency_PC1	-0.035205	0.261223	-0.135	0.8928	1.569
Stock_PC1	0.006085	0.114543	0.053	0.9576	1.688
GoldOils_PC1	0.105484	0.144950	0.728	0.4669	1.556
Bitcoin_return	0.623207	0.112654	5.532	3.71×10^{-8}	3.080
Ethereum_return	-0.200440	0.102357	-1.958	0.0504	4.109
EthereumClassic_return	0.370332	0.060264	6.145	1.01×10^{-9}	2.191
Tether_return	0.705426	0.745820	0.946	0.3444	1.023
XRP_return	0.077039	0.057647	1.336	0.1816	1.990
Cardano_return	0.172873	0.071613	2.414	0.0159	2.791
Model Statistics					
Residual standard error: 10.75 on 1554 degrees of freedom					
Multiple R-squared: 0.1765					
Adjusted R-squared: 0.1718					
F-statistic: 37.02 on 9 and 1554 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 113.7217, Standard deviation 145.0056					

in the model. The bootstrap average prediction error is 113.7217, modestly lower than the 122.8599 from the full model, suggesting a notable improvement in predictive accuracy.

4.4.2 Models for Next-One-Day Returns

Following the models for the present day's returns, we now develop regression models aimed at forecasting the next-day return for each of seven major cryptocurrencies. In contrast to the present-day models, this framework utilizes lagged predictors, specifically, the previous day's returns of other cryptocurrencies and principal components from the previous day derived from currency pairs, stock indices, and commodity indices. The model, for example, for Bitcoin is expressed as,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \varepsilon \tag{4.7}$$

Where

- Y: Daily return of Bitcoin on day t
- X_1: Currency_PC1 value of day t-1
- X_2: Stock_PC1 value of day t-1
- X_3: GoldOils_PC1 value of day t-1
- X_4: Daily return of Bitcoin on day t-1
- X_5: Daily return of Ethereum on day t-1
- X_6: Daily return of Ethereum Classic on day t-1
- X_7: Daily return of Tether on day t-1
- X_8: Daily return of XRP on day t-1
- X_9: Daily return of Cardano on day t-1
- X_{10}: Daily return of Dogecoin on day t-1
- $\beta_0, \ldots, \beta_{10}$: Unknown Regression parameters
- ε: random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{1563\times1} = \mathbf{X}_{1563\times11}\beta_{11\times1} + \varepsilon_{1563\times1},\ \ Rank(\mathbf{X}) = 11 < 1563 \tag{4.8}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin returns on day t, $\mathbf{X}$ is the design matrix of predictors containing 10 predictors of lagged returns on day $t-1$ and the intercept, β is the vector of unknown parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

As presented in Tables 4.56 - 4.62, the Multiple R^2 values for all models are very low, ranging from 0.009 (Bitcoin) to 0.1358 (Tether). This indicates that the models explain less than 14% of the variability in next-day returns, with most explaining under 5%. These values are substantially lower than those observed in the present-day models from the previous section. The bootstrap average prediction errors are relatively high for most cryptocurrencies, especially for Dogecoin (130.60) and Ethereum Classic (44.42). Even for more stable assets like Tether, the prediction error (0.1193) is only marginally better than the full model (See Table 4.35). Across all models, few predictors are statistically significant. Occasionally, lagged returns of Tether, Bitcoin, Ethereum Classic or Dogecoin show significance, but these effects are inconsistent and weak. The principal components derived from macroeconomic indicators also fail to provide meaningful predictive power. These results collectively suggest that next-day return prediction is substantially more difficult than same-day modeling. The temporal lag appears to break the strong intra-cryptocurrency relationships observed in the previous section. Moreover, the volatility and noise inherent in cryptocurrency markets likely overwhelm any signal that could be captured by lagged financial indicators or macroeconomic trends.

Table 4.56: Model Statistics for Bitcoin Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.180387	0.107173	1.683	0.0925	
Bitcoin_return	0.012546	0.044772	0.280	0.7793	3.140
Ethereum_return	-0.040119	0.040335	-0.995	0.3201	4.119
EthereumClassic_return	-0.012897	0.024011	-0.537	0.5913	2.245
Tether_return	0.650189	0.293616	2.214	0.0269	1.024
XRP_return	-0.004663	0.022719	-0.205	0.8374	1.995
Cardano_return	0.029097	0.028250	1.030	0.3032	2.804
Dogecoin_return	-0.003216	0.009984	-0.322	0.7474	1.214
Currency_PC1	-0.184199	0.102810	-1.792	0.0734	1.569
Stock_PC1	-0.011898	0.045086	-0.264	0.7919	1.688
GoldOils_PC1	0.022003	0.057060	0.386	0.6998	1.557
Model Statistics					
Residual standard error: 4.229 on 1552 degrees of freedom					
Multiple R-squared: 0.009285					
Adjusted R-squared: 0.002902					
F-statistic: 1.455 on 10 and 1552 DF, p-value: 0.1508					
Bootstrap sample prediction errors: Average 18.0853, Standard deviation 2.8689					

4.5 Concluding Remarks

This chapter presented a comprehensive modeling framework for forecasting the daily returns of major cryptocurrencies using a diverse set of financial indicators and advanced regression techniques. The modeling strategy integrated classical regression techniques with modern regularization and shrinkage methods, including full model estimation, submodel selection via AIC, BIC, and LASSO, and penalized approaches such as Ridge, Elastic Net, Adaptive LASSO, and SCAD. Additionally, post-shrinkage estimators were employed to enhance robustness, particularly in the presence of weak or moderately informative predictors. Dimensionality reduction through Principal Component Analysis was used to address multicollinearity and improve interpretability by summarizing macroeconomic indicators into compact, orthogonal components.

Table 4.57: Model Statistics for Ethereum Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.2015	0.1357	1.485	0.1377	
Bitcoin_return	-0.02261	0.05669	-0.399	0.6901	3.140
Ethereum_return	0.00787	0.05108	0.154	0.8776	4.119
EthereumClassic_return	-0.03887	0.03041	-1.278	0.2013	2.245
Tether_return	1.28100	0.37180	3.444	0.0006	1.024
XRP_return	-0.01266	0.02877	-0.440	0.6600	1.995
Cardano_return	0.01205	0.03577	0.337	0.7364	2.804
Dogecoin_return	0.00876	0.01264	0.693	0.4887	1.214
Currency_PC1	-0.2634	0.1302	-2.023	0.0432	1.569
Stock_PC1	0.00007	0.05709	0.001	0.9990	1.688
GoldOils_PC1	0.02020	0.07225	0.280	0.7799	1.557
Model Statistics					
Residual standard error: 5.356 on 1552 degrees of freedom					
Multiple R-squared: 0.0154					
Adjusted R-squared: 0.009061					
F-statistic: 2.428 on 10 and 1552 DF,p-value: 0.007196					
Bootstrap sample prediction errors: Average 28.9035, Standard deviation 4.2695					

The results from present-day models revealed strong interdependencies among cryptocurrencies. Returns of assets such as Bitcoin, Ethereum, Cardano, and Dogecoin were consistently significant predictors of each other, underscoring the dominant role of internal market dynamics. In contrast, macroeconomic indicators including currency pairs, stock indices and commodity prices, exhibited limited predictive power. Even when aggregated via PCA, their influence remained secondary. The explanatory power of these models, as measured by the multiple R^2, was relatively high for most cryptocurrencies. Tether, as expected for a stablecoin, showed minimal variability and low R^2.

Prediction accuracy, evaluated through bootstrap-based prediction errors, was generally comparable or slightly improved in PCA-based models relative to full models. This suggests that dimensionality reduction did not compromise model performance and,

Table 4.58: Model Statistics for Ethereum Classic Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.183378	0.167897	1.092	0.2749	
Bitcoin_return	-0.219199	0.070140	-3.125	0.0018	3.140
Ethereum_return	-0.007554	0.063189	-0.120	0.9049	4.119
EthereumClassic_return	0.141847	0.037616	3.771	0.0002	2.245
Tether_return	1.133095	0.459978	2.463	0.0139	1.024
XRP_return	0.001504	0.035592	0.042	0.9663	1.995
Cardano_return	-0.011213	0.044256	-0.253	0.8000	2.804
Dogecoin_return	0.020802	0.015641	1.330	0.1837	1.214
Currency_PC1	-0.320269	0.161062	-1.988	0.0469	1.569
Stock_PC1	0.061870	0.070632	0.876	0.3812	1.688
GoldOils_PC1	0.037549	0.089390	0.420	0.6745	1.557
Model Statistics					
Residual standard error: 6.626 on 1552 degrees of freedom					
Multiple R-squared: 0.02207					
Adjusted R-squared: 0.01577					
F-statistic: 3.503 on 10 and 1552 DF, p-value: 0.0001389					
Bootstrap sample prediction errors: Average 44.4237, Standard deviation 8.7315					

in some cases, enhanced it. The use of shrinkage and regularization techniques contributed to improved stability and generalization across estimation strategies.

In contrast, next-day forecasting proved substantially more challenging. Models using lagged returns and lagged principal components yielded very low R^2 values, with the exception of Tether. Prediction errors were high, and few predictors were statistically significant. These findings indicate that short-term forecasting of cryptocurrency returns on a one-day-ahead basis is unreliable using linear regression frameworks and the selected financial indicators. The volatility and noise inherent in cryptocurrency markets likely obscure any predictive signal at such short time.

Table 4.59: Model Statistics for Tether Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.0002651	0.0087010	0.030	0.9757	
Bitcoin_return	0.0005021	0.0036349	0.138	0.8902	3.140
Ethereum_return	0.0001382	0.0032747	0.042	0.9663	4.119
EthereumClassic_return	-0.0016181	0.0019494	-0.830	0.4066	2.245
Tether_return	-0.3628054	0.0238375	-15.220	$< 2 \times 10^{-16}$	1.024
XRP_return	-0.0021414	0.0018445	-1.161	0.2458	1.995
Cardano_return	-0.0001005	0.0022935	-0.044	0.9650	2.804
Dogecoin_return	0.0008566	0.0008106	1.057	0.2908	1.214
Currency_PC1	0.0038228	0.0083467	0.458	0.6470	1.569
Stock_PC1	0.0035742	0.0036604	0.976	0.3290	1.688
GoldOils_PC1	-0.0080758	0.0046325	-1.743	0.0815	1.557
Model Statistics					
Residual standard error: 0.3434 on 1552 degrees of freedom					
Multiple R-squared: 0.1358					
Adjusted R-squared: 0.1303					
F-statistic: 24.4 on 10 and 1552 DF, p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 0.1193, Standard deviation 0.0443					

The chapter demonstrates that cryptocurrency return behavior is primarily driven by intra-market relationships rather than external financial indicators. While present-day models offer meaningful insights and predictive capabilities, next-day models highlight the limitations of linear approaches in capturing short-term dynamics. These results motivate the exploration of alternative modeling strategies and longer-term horizons, which will be addressed in subsequent chapters.

Table 4.60: Model Statistics for XRP Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.1496837	0.1675417	0.893	0.3718	
Bitcoin_return	-0.0006231	0.0699917	-0.009	0.9929	3.140
Ethereum_return	-0.0377809	0.0630556	-0.599	0.5491	4.119
EthereumClassic_return	-0.0803245	0.0375366	-2.140	0.0325	2.245
Tether_return	0.8308494	0.4590049	1.810	0.0705	1.024
XRP_return	0.0169225	0.0355165	0.476	0.6338	1.995
Cardano_return	0.0557674	0.0441621	1.263	0.2069	2.804
Dogecoin_return	0.0092900	0.0156080	0.595	0.5518	1.214
Currency_PC1	-0.1498056	0.1607210	-0.932	0.3514	1.569
Stock_PC1	-0.0643892	0.0704825	-0.914	0.3611	1.688
GoldOils_PC1	0.1508288	0.0892005	1.691	0.0911	1.557
Model Statistics					
Residual standard error: 6.612 on 1552 degrees of freedom					
Multiple R-squared: 0.0102					
Adjusted R-squared: 0.003825					
F-statistic: 1.6 on 10 and 1552 DF,p-value: 0.1009					
Bootstrap sample prediction errors: Average 43.8284, Standard deviation 12.9177					

Table 4.61: Model Statistics for Cardano Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.146776	0.158249	0.927	0.3538	
Bitcoin_return	-0.086769	0.066110	-1.313	0.1895	3.140
Ethereum_return	-0.049832	0.059558	-0.837	0.4029	4.119
EthereumClassic_return	0.006860	0.035455	0.193	0.8466	2.245
Tether_return	1.482714	0.433547	3.420	0.0006	1.024
XRP_return	-0.005571	0.033547	-0.166	0.8681	1.995
Cardano_return	0.057319	0.041713	1.374	0.1696	2.804
Dogecoin_return	-0.005906	0.014742	-0.401	0.6888	1.214
Currency_PC1	-0.112831	0.151807	-0.743	0.4574	1.569
Stock_PC1	-0.032965	0.066573	-0.495	0.6205	1.688
GoldOils_PC1	0.024248	0.084253	0.288	0.7735	1.557
Model Statistics					
Residual standard error: 6.245 on 1552 degrees of freedom					
Multiple R-squared: 0.01446					
Adjusted R-squared: 0.008115					
F-statistic: 2.278 on 10 and 1552 DF, p-value: 0.01203					
Bootstrap sample prediction errors: Average 39.0524, Standard deviation 4.7469					

Table 4.62: Model Statistics for Dogecoin Future Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.56030	0.29770	1.882	0.060	
Bitcoin_return	-0.22711	0.12437	-1.826	0.068	3.140
Ethereum_return	-0.05393	0.11204	-0.481	0.630	4.119
EthereumClassic_return	0.08732	0.06670	1.309	0.191	2.245
Tether_return	0.49272	0.81559	0.604	0.546	1.024
XRP_return	-0.09014	0.06311	-1.428	0.153	1.995
Cardano_return	0.08074	0.07847	1.029	0.304	2.804
Dogecoin_return	0.11064	0.02773	3.989	6.93×10^{-5}	1.214
Currency_PC1	-0.26643	0.28558	-0.933	0.351	1.569
Stock_PC1	-0.01432	0.12524	-0.114	0.909	1.688
GoldOils_PC1	-0.04940	0.15850	-0.312	0.755	1.557
Model Statistics					
Residual standard error: 11.75 on 1552 degrees of freedom					
Multiple R-squared: 0.01713					
Adjusted R-squared: 0.01079					
F-statistic: 2.704 on 10 and 1552 DF, p-value: 0.002722					
Bootstrap sample prediction errors: Average 130.6046, Standard deviation 150.8062					

CHAPTER FIVE

CRYPTOCURRENCIES: PREDICTION OF WEEKLY RETURNS

5.1 Introduction

While daily return modeling offers insights into the very short-term behavior of cryptocurrencies, weekly return predictions provide a broader temporal perspective that is often more relevant for strategic investment decisions and portfolio management. This chapter builds upon the predictive modeling framework established in Chapter 4, extending it to the weekly return horizon to assess the effectiveness of various statistical techniques in forecasting short-term trends across major cryptocurrencies.

Conrad et al. (2018) use a GARCH-MIDAS framework to show that Bitcoin's long-term volatility is significantly influenced by macroeconomic indicators such as the Baltic Dry Index and the S&P 500 volatility risk premium, with short-term volatility driven by high-frequency market dynamics . Fang et al. (2020) find that investor perceptions captured by news-based volatility indices (NVIX) play a more significant role than economic fundamentals in predicting long-term cryptocurrency volatility, with NVIX showing a robust negative impact even after controlling for global economic policy uncertainty and realized volatility . Abraham (2020) analyzes historical price data of 76 cryptocurrencies and finds that a subset of cryptocurrencies exhibit long-run cointegration and predictable price patterns. While cryptocurrency market fundamentals significantly influence prices, traditional financial fundamentals like stock indices and exchange rates are not major value drivers. Brauneis et al. (2021) analyze the liquidity of four major cryptocurrencies across four large exchanges over a four-year period and find that cryptocurrency specific factors, particularly return volatility, dollar trading volume, and number of transactions, are the most important long-term drivers of liquidity, while

general financial market variables have no explanatory power. However all these articles emphasize long-term performance, while the data are limited to only up to 2020 when the crypto market was still evolving. Our analysis have use considerably more data and we confine to only weekly returns.

The analysis begins by constructing weekly return series from daily closing prices for seven cryptocurrencies and a selection of financial indices, including major stock market benchmarks and commodity indicators. Foreign exchange indices are excluded from this weekly return analysis due to their high inter-correlation and limited influence on cryptocurrency returns, as seen in the previous chapter. To capture temporal dependencies and autocorrelations, the modeling framework incorporates both one-week and two-week lagged returns as predictors when forecasting the return of a particular crypto for the upcoming week.

Based on the dataset comprising daily closing prices for seven cryptocurrencies and eight financial indices, the following preprocessing steps were undertaken:

1. For financial indices, missing values were imputed using the most recent available data, ensuring chronological consistency.

2. To compute weekly returns, we consider the Friday to Friday time frame. The following were calculated:

   (a) Weekly return based on the Friday's closing price.

   (b) Weekly return based on the closing price two weeks prior.

The Pearson correlation matrices were calculated for these cryptocurrencies' returns and corresponding returns for various other market indices, shown in Figures 5.1 and 5.2. Across the one-week and two-weeks Pearson correlation matrices, we observe

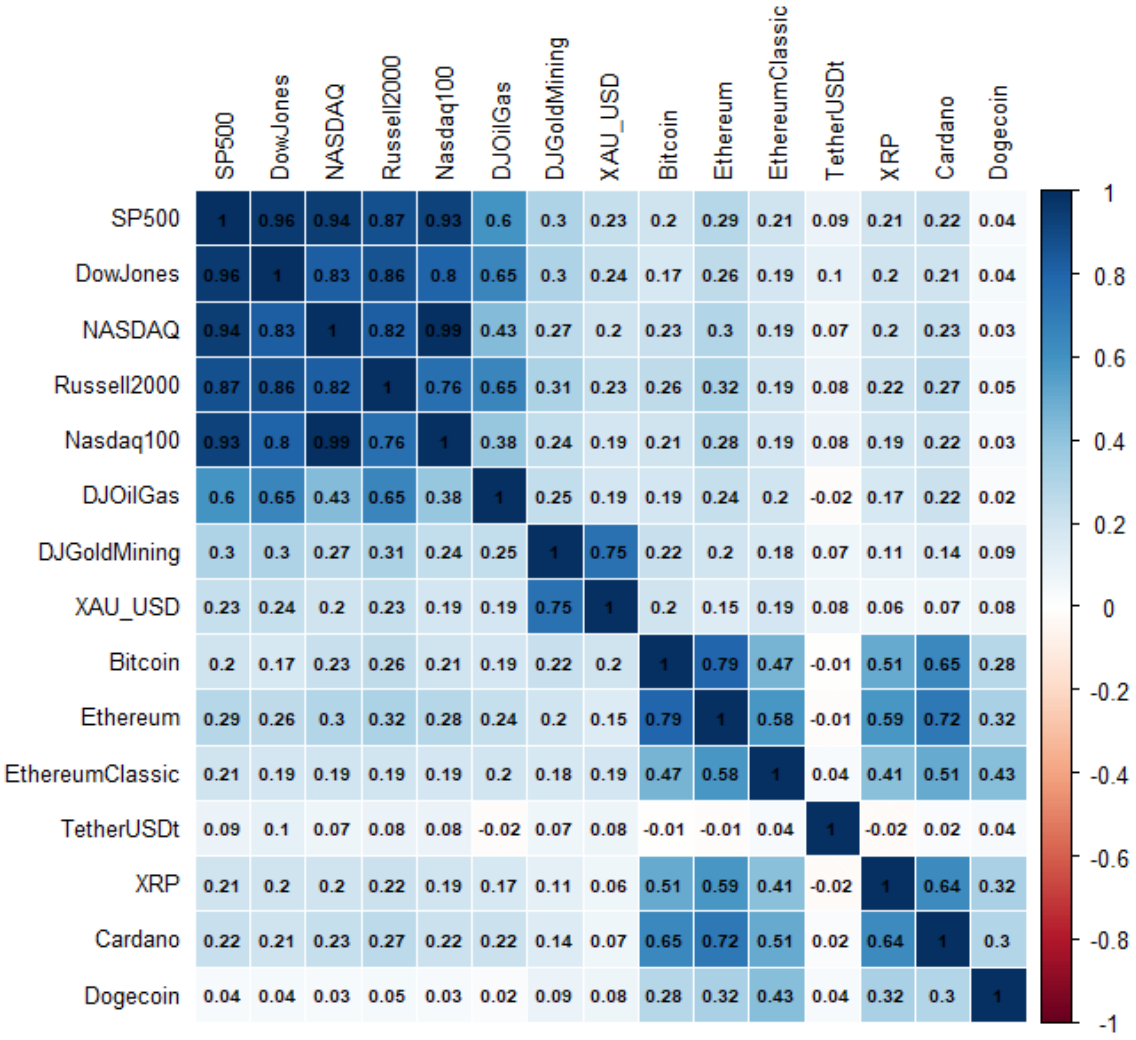


Figure 5.1: Pearson Correlation Matrix for One Week Returns

the pattern that stock indices (SP500, DowJones, NASDAQ, Russell2000, Nasdaq100) exhibit strong positive correlations with each other, reflecting their shared exposure to macroeconomic trends. Commodity indices such as DJOilGas, DJGoldMining and XAU/USD(Spot Gold price) show moderate alignment with broader markets, with GoldMining and XAU/USD often diverging due to its defensive nature. Cryptocurrencies

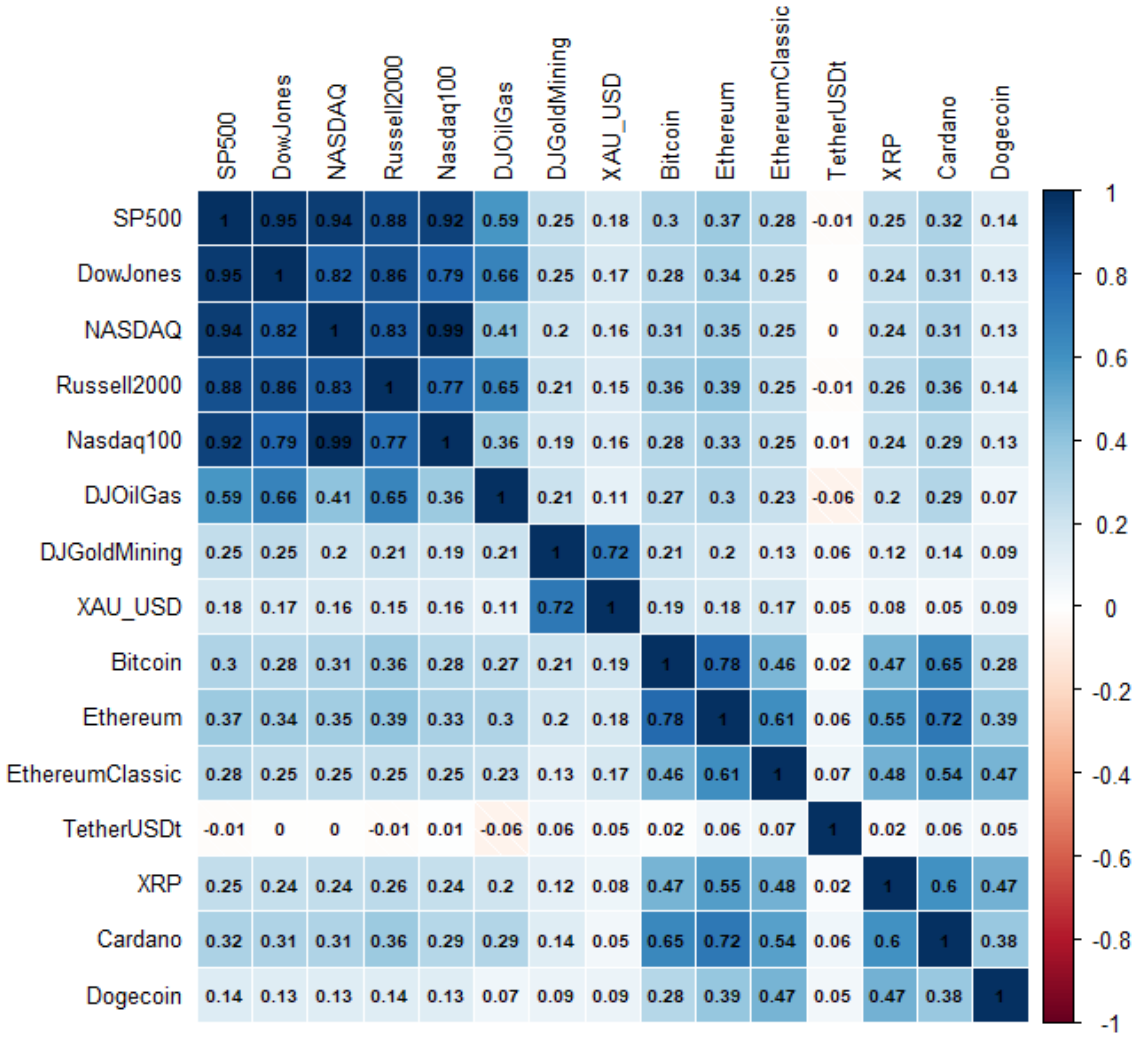


Figure 5.2: Pearson Correlation Matrix for Two Weeks Returns

exhibit moderate to strong internal correlations across all timeframes, reflecting synchronized behavior within the crypto market. An exception is Tether, which consistently shows minimal correlation with other assets due to its stablecoin nature. While correlations between cryptocurrencies and traditional financial indices remain generally low, some weak relationships do emerge. As the time horizon extends, these

correlation structures become more stable and pronounced, offering valuable insights for diversification, portfolio construction, and risk management.

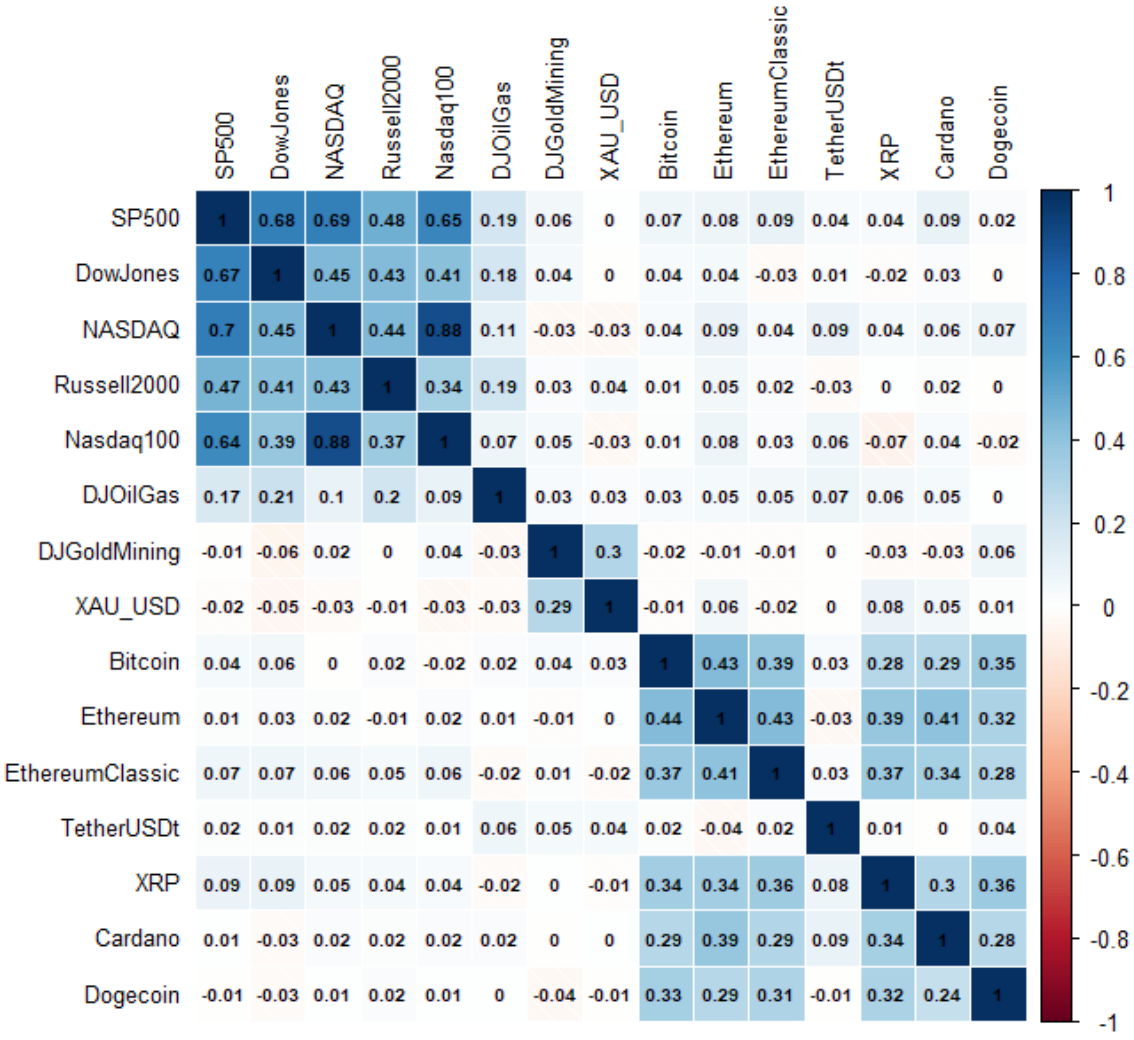


Figure 5.3: Chatterjee Correlation Matrix for One Week Returns

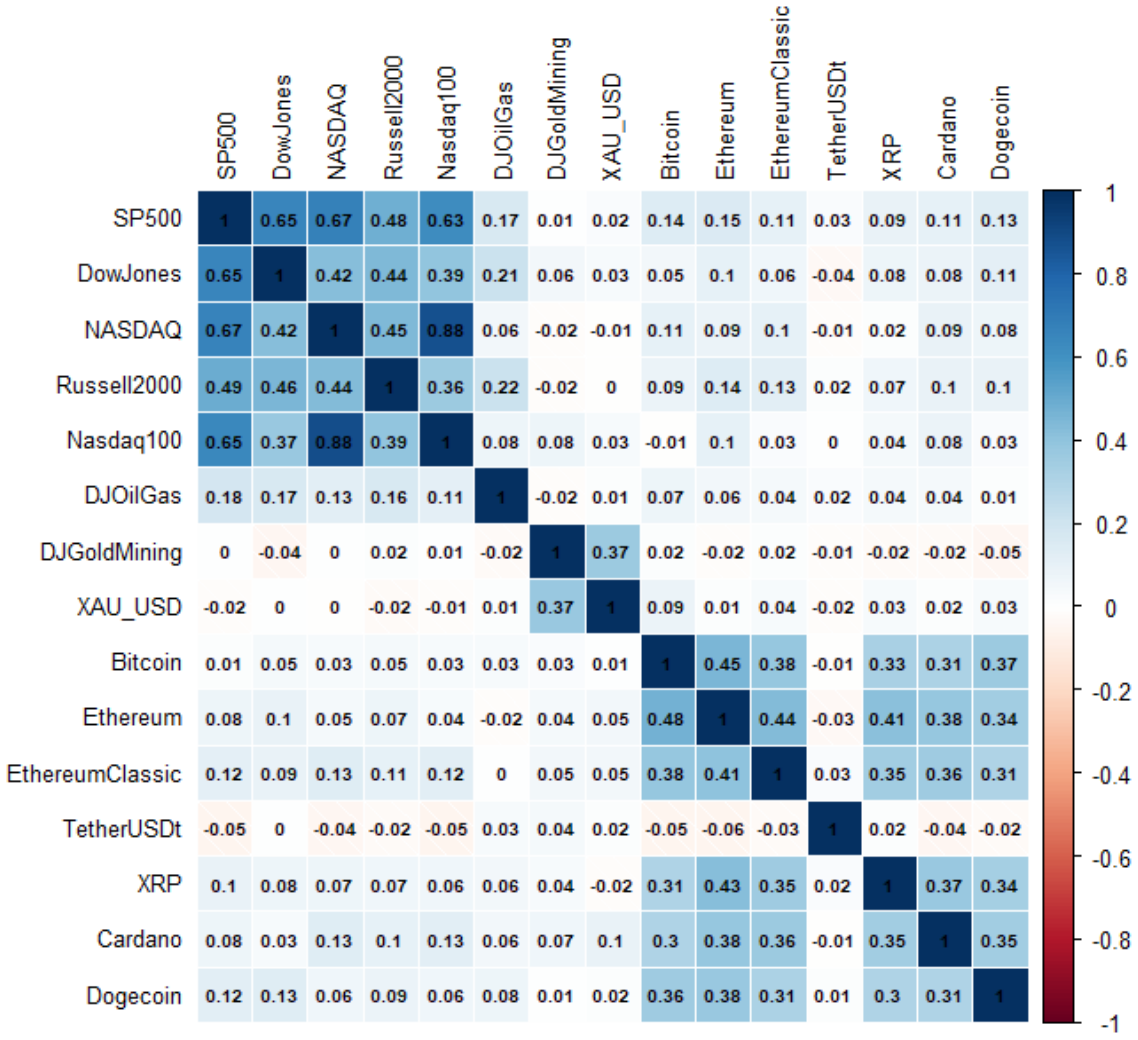


Figure 5.4: Chatterjee Correlation Matrix for Two Weeks Returns

Further, to explore nonlinear dependencies among variables, the Chatterjee correlation matrices were also computed. These are presented in Figures 5.3 and 5.4. These matrices provide a robust measure of nonlinear association, capturing both monotonic and non-monotonic relationships across cryptocurrencies and other financial indices. These reveal strong interdependencies among the weekly returns of most

cryptocurrencies, particularly between Ethereum, Bitcoin, and Ethereum Classic. Traditional financial indices such as the S&P 500, Dow Jones, NASDAQ, Russell 2000, and Nasdaq 100 exhibit high internal correlations, suggesting shared exposure to macroeconomic factors, but they show weak associations with cryptocurrencies. Commodity indices like DJ Oil & Gas, DJ Gold Mining and XAU/USD display moderate to low correlations with each other and limited influence on crypto assets. Overall, the matrix highlights that weekly crypto returns are primarily shaped by intra-market dynamics rather than broader financial market movements.

Estimators via a comprehensive suite of regression models has been employed, including the full model estimation, submodel selection via AIC, BIC and LASSO, penalized regression techniques (LASSO, Ridge, Elastic Net, SCAD, and Adaptive LASSO), and post-shrinkage estimators. To address multicollinearity and enhance interpretability, Principal Component Analysis is applied to the predictor space, with the resulting components integrated into the regression models. The analysis is organized to evaluate models for both current and next-week return predictions. Model performance is assessed using bootstrap-based prediction error metrics, enabling robust comparisons across methods and time horizons. This approach provides valuable insights into the temporal dynamics of cryptocurrency markets and supports the development of more effective forecasting strategies.

5.2 Prediction Models on Current Weekly Returns

This section presents regression models designed to predict the current weekly returns of major cryptocurrencies using the current weekly returns of key financial market indices, including those from major stock indices, commodity indices, and the other cryptocurrencies. The full model, for example, for Bitcoin is expressed as,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} + \varepsilon \tag{5.1}$$

Where

- Y: Weekly return of Bitcoin
- X_1: Weekly return of S&P 500 Index
- X_2: Weekly return of Dow Jones Index
- X_3: Weekly return of NASDAQ Index
- X_4: Weekly return of Russell 2000 Index
- X_5: Weekly return of NASDAQ 100 Index
- X_6: Weekly return of Dow Jones Oil and Gas Index
- X_7: Weekly return of Dow Jones Gold Mining Index
- X_8: Weekly return of Gold price in USD (XAU/USD)
- X_9: Weekly return of Ethereum
- X_{10}: Weekly return of Ethereum Classic
- X_{11}: Weekly return of Tether
- X_{12}: Weekly return of XRP
- X_{13}: Weekly return of Cardano
- X_{14}: Weekly return of Dogecoin

- $\beta_0, \ldots, \beta_{14}$: Unknown Regression parameters
- ε: random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{334\times 1} = \mathbf{X}_{334\times 15}\beta_{15\times 1} + \varepsilon_{334\times 1}, \;\; Rank(\mathbf{X}) = 15 < 334 \tag{5.2}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin weekly returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 15$ predictors including the intercept, β is the vector of unknown regression parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

Each model is evaluated using a full regression approach, followed by submodel selection techniques based on AIC, BIC, and LASSO criteria. To enhance robustness and mitigate overfitting, penalized regression methods such as Ridge, Elastic Net, SCAD, LASSO, and Adaptive LASSO, and post-shrinkage estimators are also applied. Model performance is assessed using bootstrap-based prediction error metrics, allowing for a rigorous comparison of accuracy and consistency across different estimation strategies.

5.2.1 A Model for Bitcoin

The full regression model for Bitcoin's weekly returns identifies several statistically significant predictors, including the S&P 500, XAU_USD, Ethereum, and Cardano, as shown in Table 5.1. However, the presence of substantial multicollinearity, evidenced by high Variance Inflation Factor (VIF) values for four predictors Nasdaq, Nasdaq 100, S&P 500 and DowJones, limits the reliability of interpreting individual

Table 5.1: Coefficients and Model Statistics for the Full Model of Bitcoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.287727	0.314470	0.915	0.36090	
SP500_1WeekReturn	-2.056754	1.040086	-1.977	0.04885	81.324
DowJones_1WeekReturn	0.468969	0.597929	0.784	0.43343	27.879
NASDAQ_1WeekReturn	1.906605	1.484914	1.284	0.20008	215.852
Russell2000_1WeekReturn	0.065699	0.305764	0.215	0.83001	12.236
Nasdaq100_1WeekReturn	-0.762318	1.267078	-0.602	0.54784	160.104
DJOilGas_1WeekReturn	0.096103	0.104844	0.917	0.36003	2.508
DJGoldMining_1WeekReturn	0.027855	0.103340	0.270	0.78768	2.422
XAU_USD_1WeekReturn	0.456416	0.241524	1.890	0.05970	2.328
Ethereum_1WeekReturn	0.507129	0.040796	12.431	$< 2 \times 10^{-16}$	2.623
EthereumClassic_1WeekReturn	-0.006291	0.020562	-0.306	0.75986	1.807
Tether_1WeekReturn	0.049824	1.007885	0.049	0.96060	1.069
XRP_1WeekReturn	0.013403	0.028086	0.477	0.63353	1.847
Cardano_1WeekReturn	0.098606	0.033492	2.944	0.00348	2.535
Dogecoin_1WeekReturn	0.004505	0.008658	0.520	0.60317	1.284
Model Statistics					
Residual standard error: 5.586 on 319 degrees of freedom					
Multiple R-squared: 0.6618					
Adjusted R-squared: 0.6469					
F-statistic: 44.58 on 14 and 319 DF					
p-value: $< 2.2 \times 10^{-16}$					

coefficients or their signs. Multicollinearity inflates the standard errors of the estimated coefficients, making it difficult to isolate the unique contribution of each predictor. Despite these limitations, the model demonstrates strong overall explanatory power, accounting for approximately 66.18% of the variance in Bitcoin's weekly returns ($R^2 = 0.6618$). The model is statistically significant (F-statistic = 44.58 on 14 and 319 degrees of freedom, p-value < 2.2×10^{-16}), indicating that the set of predictors collectively provides meaningful insight into the variability of Bitcoin's weekly returns.

To improve model interpretability and reduce overfitting, submodel selection techniques were applied using AIC, BIC, and LASSO criteria. As presented in Table 5.2,

Table 5.2: Selected Variables for Submodels on Bitcoin

Criterion	Variables Selected
BIC	XAU_USD, Ethereum, Cardano
AIC	SP500, NASDAQ, Nasdaq100, XAU_USD, Ethereum, Cardano
LASSO	SP500, DowJones, NASDAQ, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Ethereum, EthereumClassic, XRP, Cardano, Dogecoin

the BIC-selected submodel retained XAU_USD, Ethereum and Cardano as predictors, while the AIC-selected submodel included SP500, NASDAQ, Nasdaq100, XAU_USD, Ethereum, and Cardano. The LASSO method selected a broader set of predictors, including multiple financial indices and cryptocurrencies.

Table 5.3: Average and Standard Deviation of Prediction Errors for Bitcoin Weekly Return

Model	PE Average	PE s.d
Full Model	37.97662	21.414893
Submodel BIC	31.79138	6.158056
Shrinkage Estimator BIC	35.23373	14.840361
Positive Shrinkage Estimator BIC	35.23362	14.840385
Submodel AIC	31.41783	6.175870
Shrinkage Estimator AIC	35.66834	16.442310
Positive Shrinkage Estimator AIC	35.65880	16.443918
Submodel LASSO	37.81305	21.258725
Shrinkage Estimator LASSO	133.78730	612.561307
Positive Shrinkage Estimator LASSO	133.78730	612.561307
LASSO	37.86345	21.325732
ENET	37.79918	21.186052
ALASSO	37.84813	21.770392
SCAD	38.19638	23.615943
RIDGE	37.60468	19.856217

Model performance was evaluated using bootstrap-based prediction error metrics. The Table 5.3 shows that the AIC-selected submodel achieved the lowest average prediction error and standard deviation, indicating superior predictive accuracy and consistency. In contrast, the Shrinkage Estimator LASSO and Positive Shrinkage Estimator LASSO models exhibit very high prediction errors and variability, reflecting poor accuracy and instability. Adding too many unnecessary terms in the model, as done for Lasso method, apparently adversely affects the prediction.

5.2.2 A Model for Ethereum

Table 5.4: Coefficients and Model Statistics for the Full Model of Ethereum

Variable	**Estimate**	**Std. Error**	**t value**	**P value**	**VIF**
(Intercept)	0.069711	0.354677	0.197	0.84431	
SP500_1WeekReturn	1.021012	1.177377	0.867	0.38649	82.127
DowJones_1WeekReturn	-0.329287	0.673934	-0.489	0.62546	27.912
NASDAQ_1WeekReturn	0.212964	1.676956	0.127	0.89902	216.957
Russell2000_1WeekReturn	0.008747	0.344453	0.025	0.97976	12.237
Nasdaq100_1WeekReturn	-0.453824	1.427883	-0.318	0.75082	160.235
DJOilGas_1WeekReturn	-0.048417	0.118226	-0.410	0.68243	2.513
DJGoldMining_1WeekReturn	0.018655	0.116416	0.160	0.87279	2.422
XAU_USD_1WeekReturn	-0.199309	0.273356	-0.729	0.46646	2.351
Bitcoin_1WeekReturn	0.643490	0.051765	12.431	$< 2 \times 10^{-16}$	1.992
EthereumClassic_1WeekReturn	0.108684	0.022352	4.862	1.82×10^{-6}	1.683
Tether_1WeekReturn	-0.964106	1.134053	-0.850	0.39588	1.066
XRP_1WeekReturn	0.082515	0.031309	2.635	0.00881	1.809
Cardano_1WeekReturn	0.185518	0.036798	5.042	7.76×10^{-7}	2.412
Dogecoin_1WeekReturn	0.002426	0.009756	0.249	0.80375	1.285
Model Statistics					
Residual standard error: 6.292 on 319 degrees of freedom					
Multiple R-squared: 0.7431					
Adjusted R-squared: 0.7319					
F-statistic: 65.92 on 14 and 319 DF					
p-value: $< 2.2 \times 10^{-16}$					

For Ethereum's weekly returns, the full model identifies several statistically significant predictors, including Bitcoin, Ethereum Classic, XRP, and Cardano, as shown in Table 5.4 . There are predictors presenting high VIF values, such as NASDAQ, Nasdaq 100 and S&P 500, indicating substantial multicollinearity. It limits the interpretability of individual coefficients and their signs. Therefore, while some predictors are statistically significant, their specific influence on Ethereum's returns should be interpreted with caution. The model explains approximately 74.31% of the variability in Ethereum's weekly returns ($R^2 = 0.7431$) and is highly statistically significant (F-statistic: 65.92, p-value$< 2.2 \times 10^{-16}$), indicating that the set of predictors collectively provides a robust framework for understanding Ethereum's short-term return dynamics.

Table 5.5: Selected Variables for Submodels on Ethereum

Criterion	**Variables Selected**
BIC	SP500, Bitcoin, EthereumClassic, XRP, Cardano
AIC	SP500, Bitcoin, EthereumClassic, XRP, Cardano
LASSO	SP500, DowJones, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Bitcoin, EthereumClassic, Tether, XRP, Cardano, Dogecoin

Submodel selection results, summarized in Table 5.5, show that both AIC and BIC criteria selected the same subset of predictors, while the LASSO method retained a broader set of variables, including additional financial indices and cryptocurrencies.

As shown in Table 5.6, the BIC-selected submodel and the AIC-selected submodel achieved the lowest average prediction error, indicating superior predictive accuracy and consistency. The Positive Shrinkage Estimator based on BIC (or AIC) is the most reliable

Table 5.6: Average and Standard Deviation of Prediction Errors for Ethereum Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	41.01881	9.739936
Submodel BIC	39.57944	9.435414
Shrinkage Estimator BIC	39.94553	9.415689
Positive Shrinkage Estimator BIC	39.87788	9.413257
Submodel AIC	39.57944	9.435414
Shrinkage Estimator AIC	39.94553	9.415689
Positive Shrinkage Estimator AIC	39.87788	9.413257
Submodel LASSO	40.88827	9.724601
Shrinkage Estimator LASSO	114.41388	704.040054
Positive Shrinkage Estimator LASSO	114.41388	704.040054
LASSO	40.84561	9.737992
ENET	40.78614	9.736584
ALASSO	40.80897	9.647767
SCAD	41.05294	9.878419
RIDGE	40.74874	10.183732

in terms of prediction consistency, with the lowest standard deviation of prediction errors. Conversely, the LASSO-based shrinkage estimators exhibited higher prediction errors and variability.

5.2.3 A Model for Ethereum Classic

The full model for Ethereum Classic's weekly returns identifies several significant predictors, including S&P 500, Dow Jones, XAU_USD, Ethereum, Cardano, and Dogecoin, as shown in Table 5.7. There are notably high VIF values present for NASDAQ, Nasdaq 100, and the S&P 500, indicating substantial multicollinearity and limiting the interpretability of individual coefficients. The model explains approximately 44.67% of the variability in Ethereum Classic's weekly returns ($R^2 = 0.4467$) and is

Table 5.7: Coefficients and Model Statistics for the Full Model of Ethereum Classic

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.554687	0.856703	-0.647	0.5178	
SP500_1WeekReturn	7.248155	2.819882	2.570	0.0106	80.651
DowJones_1WeekReturn	-3.846120	1.615137	-2.381	0.0178	27.445
NASDAQ_1WeekReturn	-6.966545	4.034296	-1.727	0.0852	214.958
Russell2000_1WeekReturn	0.000714	0.832502	0.001	0.9993	12.237
Nasdaq100_1WeekReturn	3.772101	3.445106	1.095	0.2744	159.686
DJOilGas_1WeekReturn	0.236507	0.285506	0.828	0.4081	2.509
DJGoldMining_1WeekReturn	-0.221704	0.281102	-0.789	0.4309	2.417
XAU_USD_1WeekReturn	1.397935	0.656571	2.129	0.0340	2.321
Bitcoin_1WeekReturn	-0.046627	0.152408	-0.306	0.7599	2.956
Ethereum_1WeekReturn	0.634860	0.130568	4.862	1.82×10^{-6}	3.624
Tether_1WeekReturn	1.832717	2.742057	0.668	0.5044	1.067
XRP_1WeekReturn	0.014587	0.076486	0.191	0.8489	1.848
Cardano_1WeekReturn	0.220825	0.091581	2.411	0.0165	2.557
Dogecoin_1WeekReturn	0.122639	0.022560	5.436	1.08×10^{-7}	1.176
Model Statistics					
Residual standard error: 15.21 on 319 degrees of freedom					
Multiple R-squared: 0.4467					
Adjusted R-squared: 0.4224					
F-statistic: 18.4 on 14 and 319 DF					
p-value: $< 2.2 \times 10^{-16}$					

statistically significant overall (F-statistic: 18.4, p-value$< 2.2 \times 10^{-16}$), suggesting that the set of predictors collectively provides a meaningful framework for analyzing Ethereum Classic's weekly return behavior.

Submodel selection was conducted using AIC, BIC, and LASSO criteria, as presented in Table 5.8. Notably, the LASSO method selected all available predictors, resulting in a submodel identical to the full model. Consequently, post-shrinkage estimators could not be derived from the LASSO submodel due to the absence of variable reduction.

Table 5.8: Selected Variables for Submodels on Ethereum Classic

Criterion	**Variables Selected**
BIC	SP500, DowJones, NASDAQ, Ethereum, Cardano, Dogecoin
AIC	SP500, DowJones, NASDAQ, Nasdaq100, XAU_USD, Ethereum, Cardano, Dogecoin
LASSO	SP500, DowJones, NASDAQ, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Bitcoin, Ethereum, Tether, XRP, Cardano, Dogecoin

Table 5.9: Average and Standard Deviation of Prediction Errors for Ethereum Classic Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	296.1885	312.1959
Submodel BIC	285.0711	296.8225
Shrinkage Estimator BIC	290.2620	306.7324
Positive Shrinkage Estimator BIC	290.2888	306.8049
Submodel AIC	279.4496	280.3443
Shrinkage Estimator AIC	290.5093	304.0668
Positive Shrinkage Estimator AIC	290.4028	304.2892
Submodel LASSO	296.1885	312.1959
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	295.9107	313.1379
ENET	295.5176	313.2507
ALASSO	295.8327	313.1036
SCAD	302.1284	332.4862
RIDGE	283.5889	281.0447

As summarized in Table 5.9, The Submodel based on AIC is the most accurate for predicting Ethereum Classic's weekly return, with the lowest average prediction error. It also shows relatively good consistency with a standard deviation of prediction errors. The

RIDGE model has a slightly higher average prediction error, but still performs well. Conversely, the SCAD model has the highest average prediction error, indicating the least accuracy among the models.

5.2.4 A Model for Tether

Table 5.10: Coefficients and Model Statistics for the Full Model of Tether

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.006536	0.017488	-0.374	0.70885	
SP500_1WeekReturn	-0.047422	0.058070	-0.817	0.41475	82.149
DowJones_1WeekReturn	0.042848	0.033161	1.292	0.19725	27.788
NASDAQ_1WeekReturn	-0.199466	0.081944	-2.434	0.01547	213.011
Russell2000_1WeekReturn	0.040607	0.016834	2.412	0.01642	12.018
Nasdaq100_1WeekReturn	0.183513	0.069674	2.634	0.00885	156.874
DJOilGas_1WeekReturn	-0.012542	0.005789	-2.166	0.03103	2.478
DJGoldMining_1WeekReturn	0.001697	0.005741	0.296	0.76769	2.422
XAU_USD_1WeekReturn	0.008245	0.013484	0.611	0.54135	2.352
Bitcoin_1WeekReturn	0.000154	0.003110	0.049	0.96060	2.956
Ethereum_1WeekReturn	-0.002345	0.002758	-0.850	0.39588	3.884
EthereumClassic_1WeekReturn	0.000763	0.001142	0.668	0.50438	1.805
XRP_1WeekReturn	-0.001626	0.001558	-1.043	0.29755	1.842
Cardano_1WeekReturn	0.001906	0.001883	1.012	0.31207	2.596
Dogecoin_1WeekReturn	0.000178	0.000481	0.369	0.71231	1.285
Model Statistics					
Residual standard error: 0.3103 on 319 degrees of freedom					
Multiple R-squared: 0.0642					
Adjusted R-squared: 0.02313					
F-statistic: 1.563 on 14 and 319 DF					
p-value: 0.08806					

The full regression model for Tether's weekly returns reveals limited explanatory power, as presented in Table 5.10. Among the financial indices, NASDAQ, Russell 2000, Nasdaq 100, and DJ Oil & Gas exhibit statistical significance. This may be due to high multicollinearity as evident in high VIF values and arbitrarily positive and negative signs of estimated coefficients. None of the other cryptocurrency predictors demonstrate any meaningful influence. This outcome aligns with Tether's design as a stablecoin, which inherently exhibits minimal volatility and weak correlation with broader market movements. The model explains only a small portion of the variability in Tether's weekly returns ($R^2 = 0.0642$). The overall model is marginally significant (F-statistic = 1.563 on 14 and 319 degrees of freedom, p-value = 0.08806), suggesting that while some predictors contribute modestly, the model lacks strong predictive capability.

Table 5.11: Selected Variables for Submodels on Tether

Criterion	**Variables Selected**
BIC	NASDAQ, Russell2000, Nasdaq100
AIC	NASDAQ, Russell2000, Nasdaq100, DJOilGas
LASSO	No predictors selected

For the submodel selection results, the BIC-selected submodel retained NASDAQ, Russell 2000, and Nasdaq 100 as predictors, while the AIC-selected submodel additionally included DJ Oil & Gas, the LASSO method did not select any predictors, as shown in Table 5.11.

Model performance was evaluated using bootstrap-based prediction error metrics. As shown in Table 5.12, the AIC-selected submodel achieved the lowest average

Table 5.12: Average and Standard Deviation of Prediction Errors for Tether Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	0.10391842	0.03014983
Submodel BIC	0.09985529	0.02874994
Shrinkage Estimator BIC	0.10039078	0.02806886
Positive Shrinkage Estimator BIC	0.10033518	0.02810292
Submodel AIC	0.09876127	0.02790454
Shrinkage Estimator AIC	0.10029411	0.02780888
Positive Shrinkage Estimator AIC	0.10014662	0.02782559
Submodel LASSO	-	-
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	0.10177844	0.03095479
ENET	0.10177844	0.03095479
ALASSO	0.10177844	0.03095479
SCAD	0.10253760	0.03074421
RIDGE	0.10386882	0.03113545

prediction error and standard deviation, indicating the best predictive accuracy and consistency among all models. The full model exhibited the highest prediction error, while penalized methods such as LASSO, ALASSO, and Elastic Net offered moderate improvements. These findings reinforce the notion that Tether's weekly returns are largely stable and weakly associated with both financial indices and other cryptocurrencies, limiting the effectiveness of traditional regression-based forecasting approaches.

5.2.5 A Model for XRP

The full regression model for XRP's weekly returns identifies Ethereum, Cardano, and Dogecoin as statistically significant predictors, as shown in Table 5.13. The presence of high VIF values for NASDAQ, Nasdaq 100, and S&P 500 indicates substantial multicollinearity, which undermines the reliability of individual coefficient estimates. The

Table 5.13: Coefficients and Model Statistics for the Full Model of XRP

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.284458	0.627295	-0.453	0.65052	
SP500_1WeekReturn	-0.616733	2.085062	-0.296	0.76759	82.298
DowJones_1WeekReturn	1.040664	1.191277	0.874	0.38301	27.866
NASDAQ_1WeekReturn	-1.315675	2.965858	-0.444	0.65763	216.834
Russell2000_1WeekReturn	0.026978	0.609369	0.044	0.96471	12.237
Nasdaq100_1WeekReturn	1.243790	2.525505	0.492	0.62271	160.164
DJOilGas_1WeekReturn	-0.070712	0.209170	-0.338	0.73554	2.513
DJGoldMining_1WeekReturn	-0.003150	0.205960	-0.015	0.98781	2.422
XAU_USD_1WeekReturn	-0.162233	0.483911	-0.335	0.73765	2.354
Bitcoin_1WeekReturn	0.053228	0.111536	0.477	0.63353	2.954
Ethereum_1WeekReturn	0.258247	0.097989	2.635	0.00881	3.810
EthereumClassic_1WeekReturn	0.007815	0.040980	0.191	0.84887	1.807
Tether_1WeekReturn	-2.092145	2.005103	-1.043	0.29755	1.065
Cardano_1WeekReturn	0.429228	0.063230	6.788	5.54×10^{-11}	2.275
Dogecoin_1WeekReturn	0.041992	0.017100	2.456	0.01460	1.262
Model Statistics					
Residual standard error: 11.13 on 319 degrees of freedom					
Multiple R-squared: 0.459					
Adjusted R-squared: 0.4353					
F-statistic: 19.34 on 14 and 319 DF					
p-value: $< 2.2 \times 10^{-16}$					

model explains approximately 45.9% of the variability in XRP's weekly returns ($R^2 = 0.459$), with a highly significant overall F-statistic ($F = 19.34$, $p < 2.2 \times 10^{-16}$), indicating moderate explanatory power.

As presented in Table 5.14, both AIC and BIC selected the same subset of predictors (Ethereum, Cardano, and Dogecoin), highlighting their consistent relevance across selection methods. In contrast, the LASSO method retained all explanatory variables, resulting in a submodel identical to the full model. Consequently, post-shrinkage estimators could not be derived from the LASSO submodel due to the absence of variable reduction.

Table 5.14: Selected Variables for Submodels on XRP

Criterion	**Variables Selected**
BIC	Ethereum, Cardano, Dogecoin
AIC	Ethereum, Cardano, Dogecoin
LASSO	SP500, DowJones, NASDAQ, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Bitcoin, Ethereum, EthereumClassic, Tether, Cardano, Dogecoin

Table 5.15: Average and Standard Deviation of Prediction Errors for XRP Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	130.6080	58.95107
Submodel BIC	121.4574	57.40544
Shrinkage Estimator BIC	124.4791	57.58755
Positive Shrinkage Estimator BIC	124.4528	57.59019
Submodel AIC	121.4574	57.40544
Shrinkage Estimator AIC	124.4791	57.58755
Positive Shrinkage Estimator AIC	124.4528	57.59019
Submodel LASSO	130.6080	58.95107
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	130.1733	58.87402
ENET	129.9474	58.85178
ALASSO	130.2534	58.86616
SCAD	126.1519	57.73346
RIDGE	126.4940	59.21795

The AIC and BIC submodels yielded the lowest average prediction error and standard deviation, indicating superior predictive accuracy and consistency, as shown in Table 5.15. In comparison, post-shrinkage estimators based on the AIC and BIC submodels performed slightly worse, while penalized methods such as Ridge and SCAD

offered moderate improvements over the full model. These findings emphasize the importance of focused variable selection and suggest that XRP's weekly returns are best explained by a small set of influential cryptocurrency predictors, with limited contribution from broader financial indices.

5.2.6 A Model for Cardano

Table 5.16: Coefficients and Model Statistics for the Full Model of Cardano

Variable	**Estimate**	**Std. Error**	**t value**	**P value**	**VIF**
(Intercept)	0.002946	0.519391	0.006	0.99548	
SP500_1WeekReturn	-1.848112	1.722978	-1.073	0.28425	82.025
DowJones_1WeekReturn	0.441267	0.986913	0.447	0.65509	27.916
NASDAQ_1WeekReturn	0.940521	2.455090	0.383	0.70191	216.868
Russell2000_1WeekReturn	0.224157	0.504232	0.445	0.65695	12.230
Nasdaq100_1WeekReturn	-0.008194	2.091203	-0.004	0.99688	160.286
DJOilGas_1WeekReturn	0.217228	0.172738	1.258	0.20947	2.502
DJGoldMining_1WeekReturn	0.177957	0.170186	1.046	0.29651	2.414
XAU_USD_1WeekReturn	-0.808507	0.398047	-2.031	0.04307	2.324
Bitcoin_1WeekReturn	0.268284	0.091123	2.944	0.00348	2.878
Ethereum_1WeekReturn	0.397791	0.078903	5.042	7.76×10^{-7}	3.606
EthereumClassic_1WeekReturn	0.081060	0.033617	2.411	0.01646	1.775
Tether_1WeekReturn	1.680555	1.659827	1.012	0.31207	1.065
XRP_1WeekReturn	0.294072	0.043320	6.788	5.54×10^{-11}	1.615
Dogecoin_1WeekReturn	-0.000724	0.014287	-0.051	0.95961	1.285
Model Statistics					
Residual standard error: 9.214 on 319 degrees of freedom					
Multiple R-squared: 0.616					
Adjusted R-squared: 0.5991					
F-statistic: 36.55 on 14 and 319 DF					
p-value: $< 2.2 \times 10^{-16}$					

The full regression model for Cardano's weekly returns identifies several statistically significant predictors, including XAU_USD, Bitcoin, Ethereum, Ethereum Classic, and XRP, as shown in Table 5.16. The high VIF values indicate substantial multicollinearity, which compromises the reliability of individual coefficient estimates. So while some predictors are statistically significant, their specific influence on Cardano's returns should be interpreted with caution. The model explains approximately 61.6% of the variability in Cardano's weekly returns ($R^2 = 0.616$), with a highly significant overall F-statistic ($F = 36.55$, $p < 2.2 \times 10^{-16}$), indicating strong explanatory power.

Table 5.17: Selected Variables for Submodels on Cardano

Criterion	**Variables Selected**
BIC	Bitcoin, Ethereum, XRP
AIC	XAU_USD, Bitcoin, Ethereum, EthereumClassic, XRP
LASSO	SP500, DowJones, NASDAQ, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Bitcoin, Ethereum, EthereumClassic, Tether, XRP, Dogecoin

Submodel selection was conducted using AIC, BIC, and LASSO criteria. As shown in Table 5.17, AIC identified a compact submodel that retained the most influential cryptocurrency predictors. BIC included additional predictors XAU_USD and EthereumClassic. While the LASSO method retained all available predictors, resulting in a submodel identical to the full model. As a result, post-shrinkage estimators could not be computed due to the lack of variable selection.

The AIC submodel achieved the lowest average prediction errors and standard deviations, demonstrating superior predictive accuracy and consistency, as presented in

Table 5.18: Average and Standard Deviation of Prediction Errors for Cardano Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	100.42761	75.59047
Submodel BIC	85.39787	26.59541
Shrinkage Estimator BIC	94.30937	58.65723
Positive Shrinkage Estimator BIC	94.28821	58.65154
Submodel AIC	84.58365	26.01382
Shrinkage Estimator AIC	95.05375	60.12794
Positive Shrinkage Estimator AIC	95.02951	60.15159
Submodel LASSO	100.42761	75.59047
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	99.99116	75.13876
ENET	99.79707	75.01804
ALASSO	100.28035	75.81760
SCAD	102.54129	80.86822
RIDGE	96.22424	65.43468

Table 5.18. The BIC submodel performed slightly worse than the AIC submodel. Post-shrinkage estimators based on AIC and BIC also offered moderate improvements over the full model. These findings underscore the importance of focused variable selection and suggest that Cardano's weekly returns are best explained by a small set of influential cryptocurrency predictors, with limited contribution from traditional financial indices.

5.2.7 A Model for Dogecoin

The full regression model for Dogecoin's weekly returns identifies Ethereum Classic and XRP as statistically significant predictors, as presented in Table 5.19. The high VIF values indicate strong multicollinearity, which compromises the reliability of

Table 5.19: Coefficients and Model Statistics for the Full Model of Dogecoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	2.622080	2.030070	1.292	0.1974	
SP500_1WeekReturn	-1.617600	6.763520	-0.239	0.8111	82.306
DowJones_1WeekReturn	1.327290	3.867980	0.343	0.7317	27.923
NASDAQ_1WeekReturn	-8.807280	9.610510	-0.916	0.3601	216.398
Russell2000_1WeekReturn	1.717900	1.974240	0.870	0.3849	12.208
Nasdaq100_1WeekReturn	6.910670	8.185810	0.844	0.3992	159.929
DJOilGas_1WeekReturn	-0.873230	0.676830	-1.290	0.1979	2.501
DJGoldMining_1WeekReturn	0.255930	0.667910	0.383	0.7018	2.421
XAU_USD_1WeekReturn	-0.144650	1.569890	-0.092	0.9266	2.354
Bitcoin_1WeekReturn	0.188250	0.361760	0.520	0.6032	2.954
Ethereum_1WeekReturn	0.079900	0.321250	0.249	0.8037	3.892
EthereumClassic_1WeekReturn	0.691330	0.127170	5.436	1.08×10^{-7}	1.654
Tether_1WeekReturn	2.404040	6.513540	0.369	0.7123	1.068
XRP_1WeekReturn	0.441800	0.179920	2.456	0.0146	1.814
Cardano_1WeekReturn	-0.011120	0.219410	-0.051	0.9596	2.604
Model Statistics					
Residual standard error: 36.11 on 319 degrees of freedom					
Multiple R-squared: 0.222					
Adjusted R-squared: 0.1879					
F-statistic: 6.503 on 14 and 319 DF					
p-value: 1.455×10^{-11}					

individual coefficient estimates. The model explains approximately 22.2% of the variability in Dogecoin's weekly returns ($R^2 = 0.222$), with a statistically significant overall fit (F-statistic = 6.503 on 14 and 319 degrees of freedom, p-value = 1.455×10^{-11}). While the explanatory power is modest compared to other cryptocurrencies, the model captures meaningful relationships within the crypto ecosystem.

As shown in Table 5.20, both AIC and BIC selected compact submodels that retained Ethereum Classic and XRP as key predictors. AIC included an additional predictor DJOilGas. In contrast, the LASSO method selected all available variables,

Table 5.20: Selected Variables for Submodels on Dogecoin

Criterion	Variables Selected
BIC	EthereumClassic, XRP
AIC	DJOilGas, EthereumClassic, XRP
LASSO	SP500, DowJones, NASDAQ, Russell2000, Nasdaq100, DJOilGas, DJGoldMining, XAU_USD, Bitcoin, Ethereum, EthereumClassic, Tether, XRP, Cardano

resulting in a submodel identical to the full model and thus post-shrinkage estimators could not be constructed due to the absence of variable reduction.

Table 5.21: Average and Standard Deviation of Prediction Errors for Dogecoin Weekly Return

Model	PE Average	PE s.d
Full Model	1386.007	1570.221
Submodel BIC	1306.126	1533.193
Shrinkage Estimator BIC	1331.487	1561.816
Positive Shrinkage Estimator BIC	1328.912	1561.533
Submodel AIC	1305.635	1501.087
Shrinkage Estimator AIC	1337.029	1544.111
Positive Shrinkage Estimator AIC	1328.944	1543.910
Submodel LASSO	1386.007	1570.221
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	1384.902	1570.678
ENET	1383.990	1570.353
ALASSO	1384.802	1570.492
SCAD	1365.908	1607.900
RIDGE	1346.341	1556.483

As indicated in Table 5.21, the AIC submodel achieved the lowest average prediction errors and standard deviations for predicting Dogecoin's weekly return, indicating superior predictive accuracy and consistency. The BIC submodel performed slightly worse than the AIC submodel. Post-shrinkage estimators based on AIC and BIC offered moderate improvements over the full model, These findings suggest that Dogecoin's weekly returns are best explained by a focused set of cryptocurrency predictors, with limited influence from traditional financial indices.

5.3 Prediction Models for Next One Week Returns

To investigate the feasibility of predicting the next week's returns for major cryptocurrencies, we construct predictive models using lagged weekly returns of cryptocurrencies and financial indices, such as major stock market benchmarks and commodity indicators. Foreign exchange indices are excluded from this analysis due to their high inter-correlation and limited influence on cryptocurrency returns, as established in prior sections. Specifically, for each day in the dataset, we use the returns from the previous one and two weeks cumulatively of eight financial indices and seven cryptocurrencies as explanatory variables. The response variable in each model is the return for the upcoming week of the target cryptocurrency.

The full model, for example, for predicting the next week's return of Bitcoin is given by,

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 \\ &+ \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} + \beta_{15} X_{15} + \beta_{16} X_{16} \\ &+ \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} + \beta_{22} X_{22} + \beta_{23} X_{23} \\ &+ \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \beta_{27} X_{27} + \beta_{28} X_{28} + \beta_{29} X_{29} + \beta_{30} X_{30} + \varepsilon \end{aligned} \tag{5.3}$$

Where

- Y: Next week's return of Bitcoin
- X_1 to X_{15}: Returns over the past one week
  - X_1: S&P 500
  - X_2: Dow Jones Index
  - X_3: NASDAQ Index
  - X_4: Russell 2000 Index
  - X_5: NASDAQ 100 Index
  - X_6: Dow Jones Oil and Gas Index
  - X_7: Dow Jones Gold Mining Index
  - X_8: Gold price in USD (XAU/USD)
  - X_9: Ethereum
  - X_{10}: Ethereum Classic
  - X_{11}: Tether
  - X_{12}: XRP
  - X_{13}: Cardano
  - X_{14}: Dogecoin
  - X_{15}: Bitcoin (past one week return)
- X_{16} to X_{30}: Cumulative returns over the two weeks prior (same assets as above)
- $\beta_0, \ldots, \beta_{30}$: Unknown regression parameters

- ε: Random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{333\times 1} = \mathbf{X}_{333\times 31}\beta_{31\times 1} + \varepsilon_{333\times 1}, \;\; Rank(\mathbf{X}) = 31 < 333 \tag{5.4}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin next week's returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 31$ predictors including the intercept, β is the vector of unknown regression parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

The modeling framework is designed to capture short-term temporal dependencies and autocorrelations, providing a practical approach for near-term forecasting. As presented in Tables 5.22 - 5.42, the empirical results indicate that the predictive power of these models is generally limited. Across all cryptocurrencies analyzed, the multiple R^2 values ranges from approximately 0.08013 (XRP) to 0.175 (Tether), suggesting that the models explain less than 18% of the variability in next-week returns. These values are substantially lower than those observed in the same-week models presented in Section 5.2, indicating that the models are unable to capture a substantial portion of the variability in next-week returns. The limited explanatory power is further reflected in the statistical insignificance of most full model estimators. In many cases, the included predictors fail to reach significance thresholds, implying that they do not adequately account for the variability in next-week performance.

The bootstrap-based prediction errors reinforce the models' limitations. Relatively high average prediction errors are observed across all assests analyzed, particularly for

highly volatile cryptocurrencies such as Dogecoin and Ethereum Classic. Even for relatively stable assets like Tether, the prediction error remains modest. These results suggest that the volatility and noise inherent in cryptocurrency markets significantly obscure predictive signal that might be derived from lagged financial indicators or macroeconomic trends. Among the various modeling strategies, the AIC-selected submodels consistently achieve lower average prediction errors across major cryptocurrencies, suggesting relatively superior predictive performance. Penalized regression methods such as SCAD and Ridge also demonstrate moderate improvements over the full model, offering more stable and parsimonious alternatives.

Despite these constraints, the analysis reveals that few predictors are statistically significant across models. While lagged returns of Tether, Bitcoin, Ethereum Classic, and Dogecoin occasionally exhibit significance, these effects are inconsistent and lack robustness. In some cases, financial indices such as Russell 2000, DJ Gold Mining, and Nasdaq 100 show statistical significance. However, these results are not uniformly observed across models. Furthermore, the presence of high VIF values for stock indices indicates multicollinearity, which compromises the reliability of coefficient estimates and their signs.

Collectively, these findings suggest that next-week return prediction using linear regression and lagged predictors is substantially more difficult than same-week modeling. The temporal lag appears to weaken the strong intra-cryptocurrency relationships observed in same-week models. Moreover, the inherent volatility and structural noise in cryptocurrency markets likely overwhelm any signal that could be captured by lagged financial or macroeconomic indicators. While informative, the findings suggest that linear regression frameworks may be inadequate for capturing the complexity of next-week

return dynamics, indicating that alternative modeling strategies or expanded predictor sets may be necessary to achieve meaningful predictive performance.

Table 5.22: Coefficients and Model Statistics for the Full Model of Bitcoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.566222	0.543895	1.041	0.2987	-
SP500_1WeekReturn	-1.071608	2.493314	-0.430	0.6677	166.637
DowJones_1WeekReturn	0.262936	1.417084	0.186	0.8529	55.868
NASDAQ_1WeekReturn	-4.364472	3.531743	-1.236	0.2175	434.489
Russell2000_1WeekReturn	1.034048	0.682098	1.516	0.1306	21.721
Nasdaq100_1WeekReturn	3.996797	3.003856	1.331	0.1843	320.096
DJOilGas_1WeekReturn	0.145042	0.259087	0.560	0.5760	5.460
DJGoldMining_1WeekReturn	-0.290359	0.255190	-1.138	0.2561	5.269
XAU_USD_1WeekReturn	-0.134377	0.601160	-0.224	0.8233	5.137
Bitcoin_1WeekReturn	0.175883	0.151881	1.158	0.2478	7.754
Ethereum_1WeekReturn	-0.161261	0.132573	-1.216	0.2248	9.874
EthereumClassic_1WeekReturn	0.027769	0.052638	0.528	0.5982	4.223
Tether_1WeekReturn	-0.606065	2.238874	-0.271	0.7868	1.882
XRP_1WeekReturn	-0.023314	0.076158	-0.306	0.7597	4.845
Cardano_1WeekReturn	0.050494	0.089686	0.563	0.5738	6.479
Dogecoin_1WeekReturn	-0.041577	0.021720	-1.914	0.0565	2.883
SP500_2WeekReturn	2.313009	1.796503	1.288	0.1989	155.321
DowJones_2WeekReturn	-0.863923	1.026327	-0.842	0.4006	52.082
NASDAQ_2WeekReturn	1.235031	2.596323	0.476	0.6346	436.782
Russell2000_2WeekReturn	-0.633782	0.529373	-1.197	0.2322	23.531
Nasdaq100_2WeekReturn	-1.638042	2.194605	-0.746	0.4560	316.615
DJOilGas_2WeekReturn	-0.158404	0.180470	-0.878	0.3808	5.535
DJGoldMining_2WeekReturn	-0.043618	0.179450	-0.243	0.8081	4.795
XAU_USD_2WeekReturn	0.548325	0.415413	1.320	0.1879	4.666
Bitcoin_2WeekReturn	0.027502	0.096550	0.285	0.7760	6.953
Ethereum_2WeekReturn	0.025113	0.084707	0.296	0.7671	9.272
EthereumClassic_2WeekReturn	-0.082374	0.036742	-2.242	0.0257	4.565
Tether_2WeekReturn	1.629170	1.770399	0.920	0.3582	1.771
XRP_2WeekReturn	-0.004474	0.048537	-0.092	0.9266	4.758
Cardano_2WeekReturn	0.020095	0.054617	0.368	0.7132	6.216
Dogecoin_2WeekReturn	0.039665	0.016913	2.345	0.0197	3.071
Model Statistics					
Residual standard error: 9.351 on 302 degrees of freedom					
Multiple R-squared: 0.1019. Adjusted R-squared: 0.01274					
F-statistic: 1.143 on 30 and 302 DF. p-value: 0.2827					

Table 5.23: Selected Variables for Submodels on Bitcoin

Criterion	Variables Selected
BIC	No predictors selected
AIC	DJGoldMining_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, XAU_USD_2WeekReturn, EthereumClassic_2WeekReturn, Dogecoin_2WeekReturn
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, XAU_USD_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, XAU_USD_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn

Table 5.24: Average and Standard Deviation of Prediction Errors for Bitcoin Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	95.16890	24.96439
Submodel BIC	-	-
Shrinkage Estimator BIC	-	-
Positive Shrinkage Estimator BIC	-	-
Submodel AIC	85.94379	18.30390
Shrinkage Estimator AIC	87.36992	20.44337
Positive Shrinkage Estimator AIC	87.34099	20.41667
Submodel LASSO	95.16890	24.96439
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	94.83954	24.74230
ENET	94.57699	24.45872
ALASSO	94.66148	24.28596
SCAD	87.65651	18.70900
RIDGE	90.56217	19.73936

Table 5.25: Coefficients and Model Statistics for the Full Model of Ethereum

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.515345	0.705493	0.730	0.4657	-
SP500_1WeekReturn	-0.839950	3.234106	-0.260	0.7953	166.637
DowJones_1WeekReturn	-0.217407	1.838115	-0.118	0.9059	55.868
NASDAQ_1WeekReturn	-7.555663	4.581064	-1.649	0.1001	434.489
Russell2000_1WeekReturn	1.742941	0.884757	1.970	0.0498	21.721
Nasdaq100_1WeekReturn	6.713183	3.896336	1.723	0.0859	320.096
DJOilGas_1WeekReturn	0.235202	0.336064	0.700	0.4845	5.460
DJGoldMining_1WeekReturn	-0.381479	0.331010	-1.152	0.2500	5.269
XAU_USD_1WeekReturn	0.198285	0.779771	0.254	0.7994	5.137
Bitcoin_1WeekReturn	0.042998	0.197007	0.218	0.8274	7.754
Ethereum_1WeekReturn	-0.126913	0.171962	-0.738	0.4611	9.874
EthereumClassic_1WeekReturn	0.114067	0.068278	1.671	0.0958	4.223
Tether_1WeekReturn	-1.125118	2.904069	-0.387	0.6987	1.882
XRP_1WeekReturn	-0.041206	0.098785	-0.417	0.6769	4.845
Cardano_1WeekReturn	0.023872	0.116332	0.205	0.8375	6.479
Dogecoin_1WeekReturn	-0.007271	0.028174	-0.258	0.7965	2.883
SP500_2WeekReturn	2.794704	2.330265	1.199	0.2313	155.321
DowJones_2WeekReturn	-0.580931	1.331260	-0.436	0.6629	52.082
NASDAQ_2WeekReturn	1.881684	3.367720	0.559	0.5768	436.782
Russell2000_2WeekReturn	-1.107515	0.686655	-1.613	0.1078	23.531
Nasdaq100_2WeekReturn	-2.571908	2.846647	-0.903	0.3670	316.615
DJOilGas_2WeekReturn	-0.261139	0.234089	-1.116	0.2655	5.535
DJGoldMining_2WeekReturn	0.020431	0.232767	0.088	0.9301	4.795
XAU_USD_2WeekReturn	0.423475	0.538836	0.786	0.4325	4.666
Bitcoin_2WeekReturn	0.088529	0.125237	0.707	0.4802	6.953
Ethereum_2WeekReturn	0.081290	0.109875	0.740	0.4600	9.272
EthereumClassic_2WeekReturn	-0.111686	0.047658	-2.343	0.0198	4.565
Tether_2WeekReturn	3.776615	2.296404	1.645	0.1011	1.771
XRP_2WeekReturn	-0.008500	0.062958	-0.135	0.8927	4.758
Cardano_2WeekReturn	0.033502	0.070845	0.473	0.6366	6.216
Dogecoin_2WeekReturn	0.023516	0.021938	1.072	0.2846	3.071
Model Statistics					
Residual standard error: 12.13 on 302 degrees of freedom					
Multiple R-squared: 0.09637. Adjusted R-squared: 0.006606					
F-statistic: 1.074 on 30 and 302 DF. p-value: 0.3676					

Table 5.26: Selected Variables for Submodels on Ethereum

Criterion	Variables Selected
BIC	No predictors selected
AIC	NASDAQ_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJGoldMining_1WeekReturn, EthereumClassic_1WeekReturn, SP500_2WeekReturn, Russell2000_2WeekReturn, XAU_USD_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, Dogecoin_2WeekReturn, Bitcoin_2WeekReturn,
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, XAU_USD_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, XAU_USD_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn

Table 5.27: Average and Standard Deviation of Prediction Errors for Ethereum Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	163.1395	40.90283
Submodel BIC	-	-
Shrinkage Estimator BIC	-	-
Positive Shrinkage Estimator BIC	-	-
Submodel AIC	144.5614	32.28640
Shrinkage Estimator AIC	148.8158	33.73295
Positive Shrinkage Estimator AIC	148.7074	33.67437
Submodel LASSO	163.1395	40.90283
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	162.6575	40.66007
ENET	162.2914	40.38776
ALASSO	161.7707	39.98695
SCAD	147.7648	36.31584
RIDGE	155.1510	36.48649

Table 5.28: Coefficients and Model Statistics for the Full Model of Ethereum Classic

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	1.089230	1.144870	0.951	0.3422	-
SP500_1WeekReturn	3.773580	5.248300	0.719	0.4727	166.637
DowJones_1WeekReturn	-2.451660	2.982890	-0.822	0.4118	55.868
NASDAQ_1WeekReturn	-4.916310	7.434140	-0.661	0.5089	434.489
Russell2000_1WeekReturn	1.137170	1.435780	0.792	0.4290	21.721
Nasdaq100_1WeekReturn	3.204400	6.322960	0.507	0.6127	320.096
DJOilGas_1WeekReturn	0.150290	0.545360	0.276	0.7831	5.460
DJGoldMining_1WeekReturn	-1.050100	0.537160	-1.955	0.0515	5.269
XAU_USD_1WeekReturn	0.665780	1.265410	0.526	0.5992	5.137
Bitcoin_1WeekReturn	0.904320	0.319700	2.829	0.0050	7.754
Ethereum_1WeekReturn	-1.003690	0.279060	-3.597	0.0004	9.874
EthereumClassic_1WeekReturn	0.116990	0.110800	1.056	0.2919	4.223
Tether_1WeekReturn	0.516960	4.712720	0.110	0.9127	1.882
XRP_1WeekReturn	0.365140	0.160310	2.278	0.0234	4.845
Cardano_1WeekReturn	0.130520	0.188780	0.691	0.4899	6.479
Dogecoin_1WeekReturn	-0.017530	0.045720	-0.383	0.7017	2.883
SP500_2WeekReturn	0.521410	3.781550	0.138	0.8904	155.321
DowJones_2WeekReturn	0.766230	2.160370	0.355	0.7231	52.082
NASDAQ_2WeekReturn	3.567250	5.465130	0.653	0.5144	436.782
Russell2000_2WeekReturn	-1.451970	1.114300	-1.303	0.1936	23.531
Nasdaq100_2WeekReturn	-3.357960	4.619530	-0.727	0.4678	316.615
DJOilGas_2WeekReturn	-0.208540	0.379880	-0.549	0.5834	5.535
DJGoldMining_2WeekReturn	0.208920	0.377730	0.553	0.5806	4.795
XAU_USD_2WeekReturn	-0.151920	0.874420	-0.174	0.8622	4.666
Bitcoin_2WeekReturn	-0.497300	0.203230	-2.447	0.0150	6.953
Ethereum_2WeekReturn	0.652080	0.178300	3.657	0.0003	9.272
EthereumClassic_2WeekReturn	-0.107560	0.077340	-1.391	0.1653	4.565
Tether_2WeekReturn	1.130960	3.726600	0.303	0.7617	1.771
XRP_2WeekReturn	-0.104930	0.102170	-1.027	0.3052	4.758
Cardano_2WeekReturn	-0.095550	0.114970	-0.831	0.4066	6.216
Dogecoin_2WeekReturn	0.016980	0.035600	0.477	0.6337	3.071
Model Statistics					
Residual standard error: 19.68 on 302 degrees of freedom					
Multiple R-squared: 0.1218. Adjusted R-squared: 0.03451					
F-statistic: 1.396 on 30 and 302 DF. p-value: 0.08705					

Table 5.29: Selected Variables for Submodels on Ethereum Classic

Criterion	Variables Selected
BIC	Bitcoin_1WeekReturn, Ethereum_1WeekReturn, XRP_1WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn
AIC	SP500_1WeekReturn, DJGoldMining_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, XRP_1WeekReturn, Russell2000_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, XRP_2WeekReturn
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, XAU_USD_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, XAU_USD_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn

Table 5.30: Average and Standard Deviation of Prediction Errors for Ethereum Classic Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	403.3480	255.3755
Submodel BIC	363.8882	261.3930
Shrinkage Estimator BIC	364.5649	253.1928
Positive Shrinkage Estimator BIC	364.3384	253.2490
Submodel AIC	356.9965	244.9314
Shrinkage Estimator AIC	363.6573	247.6826
Positive Shrinkage Estimator AIC	362.3757	247.7778
Submodel LASSO	403.3480	255.3755
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	401.8170	255.2674
ENET	400.6104	255.1202
ALASSO	401.5830	255.1305
SCAD	375.1621	281.8951
RIDGE	378.3086	255.4822

Table 5.31: Coefficients and Model Statistics for the Full Model of Tether

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.008442	0.017420	0.485	0.6283	-
SP500_1WeekReturn	0.000759	0.079840	0.010	0.9924	166.637
DowJones_1WeekReturn	0.002364	0.045380	0.052	0.9585	55.868
NASDAQ_1WeekReturn	0.204900	0.113100	1.812	0.0711	434.489
Russell2000_1WeekReturn	-0.023620	0.021840	-1.081	0.2804	21.721
Nasdaq100_1WeekReturn	-0.184300	0.096190	-1.915	0.0564	320.096
DJOilGas_1WeekReturn	-0.001376	0.008297	-0.166	0.8684	5.460
DJGoldMining_1WeekReturn	0.004332	0.008172	0.530	0.5964	5.269
XAU_USD_1WeekReturn	-0.015770	0.019250	-0.819	0.4133	5.137
Bitcoin_1WeekReturn	-0.002974	0.004864	-0.612	0.5413	7.754
Ethereum_1WeekReturn	0.000279	0.004245	0.066	0.9477	9.874
EthereumClassic_1WeekReturn	0.000811	0.001686	0.481	0.6309	4.223
Tether_1WeekReturn	-0.119100	0.071690	-1.661	0.0977	1.882
XRP_1WeekReturn	0.001609	0.002439	0.660	0.5099	4.845
Cardano_1WeekReturn	0.004149	0.002872	1.445	0.1496	6.479
Dogecoin_1WeekReturn	-0.000155	0.000695	-0.223	0.8234	2.883
SP500_2WeekReturn	0.008995	0.057530	0.156	0.8759	155.321
DowJones_2WeekReturn	-0.023930	0.032870	-0.728	0.4671	52.082
NASDAQ_2WeekReturn	-0.082140	0.083140	-0.988	0.3240	436.782
Russell2000_2WeekReturn	0.013590	0.016950	0.802	0.4235	23.531
Nasdaq100_2WeekReturn	0.067660	0.070280	0.963	0.3365	316.615
DJOilGas_2WeekReturn	-0.000894	0.005779	-0.155	0.8772	5.535
DJGoldMining_2WeekReturn	-0.000617	0.005746	-0.107	0.9145	4.795
XAU_USD_2WeekReturn	0.003697	0.013300	0.278	0.7813	4.666
Bitcoin_2WeekReturn	-0.002070	0.003092	-0.670	0.5036	6.953
Ethereum_2WeekReturn	0.003124	0.002713	1.152	0.2504	9.272
EthereumClassic_2WeekReturn	-0.000622	0.001177	-0.528	0.5976	4.565
Tether_2WeekReturn	-0.148600	0.056690	-2.621	0.0092	1.771
XRP_2WeekReturn	-0.000254	0.001554	-0.163	0.8705	4.758
Cardano_2WeekReturn	-0.002099	0.001749	-1.200	0.2311	6.216
Dogecoin_2WeekReturn	-0.000046	0.000542	-0.085	0.9321	3.071
Model Statistics					
Residual standard error: 0.2994 on 302 degrees of freedom					
Multiple R-squared: 0.175. Adjusted R-squared: 0.09303					
F-statistic: 2.135 on 30 and 302 DF. p-value: 0.0007529					

Table 5.32: Selected Variables for Submodels on Tether

Criterion	Variables Selected
BIC	DowJones_2WeekReturn, Tether_2WeekReturn, Ethereum_1WeekReturn
AIC	NASDAQ_1WeekReturn, Nasdaq100_1WeekReturn, Tether_1WeekReturn, Bitcoin_2WeekReturn, Cardano_1WeekReturn, DowJones_2WeekReturn, Ethereum_2WeekReturn, Tether_2WeekReturn, Cardano_2WeekReturn
LASSO	No predictors selected

Table 5.33: Average and Standard Deviation of Prediction Errors for Tether Weekly Return

Model	PE Average	PE s.d
Full Model	0.09716215	0.02923323
Submodel BIC	0.90867050	0.60340353
Shrinkage Estimator BIC	0.37534865	0.32340833
Positive Shrinkage Estimator BIC	0.37045960	0.31206390
Submodel AIC	0.08627758	0.02482893
Shrinkage Estimator AIC	0.08788312	0.02469793
Positive Shrinkage Estimator AIC	0.08751123	0.02451183
Submodel LASSO	-	-
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	0.09804978	0.02866534
ENET	0.09815480	0.02889764
ALASSO	0.09802114	0.02855082
SCAD	0.09048631	0.02562554
RIDGE	0.09078511	0.02541571

Table 5.34: Coefficients and Model Statistics for the Full Model of XRP

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.264932	0.864804	0.306	0.7595	-
SP500_1WeekReturn	-5.709474	3.964416	-1.440	0.1509	166.637
DowJones_1WeekReturn	0.995027	2.253189	0.442	0.6591	55.868
NASDAQ_1WeekReturn	3.907865	5.615537	0.696	0.4870	434.489
Russell2000_1WeekReturn	0.830352	1.084548	0.766	0.4445	21.721
Nasdaq100_1WeekReturn	-0.824905	4.776187	-0.173	0.8630	320.096
DJOilGas_1WeekReturn	0.261805	0.411953	0.636	0.5256	5.460
DJGoldMining_1WeekReturn	-0.376643	0.405757	-0.928	0.3540	5.269
XAU_USD_1WeekReturn	0.744838	0.955855	0.779	0.4365	5.137
Bitcoin_1WeekReturn	-0.123856	0.241494	-0.513	0.6084	7.754
Ethereum_1WeekReturn	-0.106823	0.210793	-0.507	0.6127	9.874
EthereumClassic_1WeekReturn	-0.003928	0.083696	-0.047	0.9626	4.223
Tether_1WeekReturn	-2.996233	3.559851	-0.842	0.4006	1.882
XRP_1WeekReturn	0.157471	0.121092	1.300	0.1944	4.845
Cardano_1WeekReturn	0.076429	0.142602	0.536	0.5924	6.479
Dogecoin_1WeekReturn	-0.020660	0.034536	-0.598	0.5501	2.883
SP500_2WeekReturn	5.915296	2.856474	2.071	0.0392	155.321
DowJones_2WeekReturn	-1.815340	1.631879	-1.112	0.2668	52.082
NASDAQ_2WeekReturn	-4.607484	4.128202	-1.116	0.2653	436.782
Russell2000_2WeekReturn	-0.374258	0.841712	-0.445	0.6569	23.531
Nasdaq100_2WeekReturn	1.623677	3.489462	0.465	0.6420	316.615
DJOilGas_2WeekReturn	-0.260501	0.286950	-0.908	0.3647	5.535
DJGoldMining_2WeekReturn	0.172834	0.285330	0.606	0.5451	4.795
XAU_USD_2WeekReturn	-0.398545	0.660514	-0.603	0.5467	4.666
Bitcoin_2WeekReturn	0.132275	0.153517	0.862	0.3896	6.953
Ethereum_2WeekReturn	0.064091	0.134686	0.476	0.6345	9.272
EthereumClassic_2WeekReturn	-0.061787	0.058420	-1.058	0.2911	4.565
Tether_2WeekReturn	4.157972	2.814967	1.477	0.1407	1.771
XRP_2WeekReturn	-0.036378	0.077175	-0.471	0.6377	4.758
Cardano_2WeekReturn	-0.096814	0.086843	-1.115	0.2658	6.216
Dogecoin_2WeekReturn	0.055896	0.026891	2.079	0.0385	3.071
Model Statistics					
Residual standard error: 14.87 on 302 degrees of freedom					
Multiple R-squared: 0.08013. Adjusted R-squared: -0.01125					
F-statistic: 0.8769 on 30 and 302 DF. p-value: 0.6558					

Table 5.35: Selected Variables for Submodels on XRP

Criterion	Variables Selected
BIC	No predictors selected
AIC	SP500_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, SP500_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, EthereumClassic_2WeekReturn, Dogecoin_2WeekReturn
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn, XAU_USD_1WeekReturn, XAU_USD_2WeekReturn

Table 5.36: Average and Standard Deviation of Prediction Errors for XRP Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	261.8197	100.94829
Submodel BIC	-	-
Shrinkage Estimator BIC	-	-
Positive Shrinkage Estimator BIC	-	-
Submodel AIC	224.3276	71.68510
Shrinkage Estimator AIC	233.7743	80.73729
Positive Shrinkage Estimator AIC	233.7698	80.75553
Submodel LASSO	260.5903	101.51621
Shrinkage Estimator LASSO	10555.3904	105975.67482
Positive Shrinkage Estimator LASSO	10555.3904	105975.67482
LASSO	260.7664	100.39846
ENET	259.9064	99.87004
ALASSO	260.0090	99.66648
SCAD	225.2870	84.27302
RIDGE	245.9537	86.25433

Table 5.37: Coefficients and Model Statistics for the Full Model of Cardano

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.209986	0.829166	0.253	0.8002	-
SP500_1WeekReturn	-4.815335	3.801046	-1.267	0.2062	166.637
DowJones_1WeekReturn	1.809940	2.160338	0.838	0.4028	55.868
NASDAQ_1WeekReturn	-2.291565	5.384126	-0.426	0.6707	434.489
Russell2000_1WeekReturn	1.243118	1.039855	1.195	0.2328	21.721
Nasdaq100_1WeekReturn	4.285181	4.579364	0.936	0.3501	320.096
DJOilGas_1WeekReturn	0.044316	0.394977	0.112	0.9107	5.460
DJGoldMining_1WeekReturn	-0.457613	0.389036	-1.176	0.2404	5.269
XAU_USD_1WeekReturn	-0.069075	0.916465	-0.075	0.9400	5.137
Bitcoin_1WeekReturn	0.074016	0.231542	0.320	0.7494	7.754
Ethereum_1WeekReturn	-0.205216	0.202107	-1.015	0.3107	9.874
EthereumClassic_1WeekReturn	0.146780	0.080247	1.829	0.0684	4.223
Tether_1WeekReturn	-3.397182	3.413153	-0.995	0.3204	1.882
XRP_1WeekReturn	0.007403	0.116102	0.064	0.9492	4.845
Cardano_1WeekReturn	0.002448	0.136726	0.018	0.9857	6.479
Dogecoin_1WeekReturn	-0.055665	0.033113	-1.681	0.0938	2.883
SP500_2WeekReturn	2.257129	2.738761	0.824	0.4105	155.321
DowJones_2WeekReturn	-0.446354	1.564630	-0.285	0.7756	52.082
NASDAQ_2WeekReturn	1.999574	3.958082	0.505	0.6138	436.782
Russell2000_2WeekReturn	-0.887087	0.807026	-1.099	0.2726	23.531
Nasdaq100_2WeekReturn	-2.844586	3.345664	-0.850	0.3959	316.615
DJOilGas_2WeekReturn	-0.114956	0.275125	-0.418	0.6764	5.535
DJGoldMining_2WeekReturn	0.181689	0.273571	0.664	0.5071	4.795
XAU_USD_2WeekReturn	0.144679	0.633295	0.228	0.8194	4.666
Bitcoin_2WeekReturn	0.074719	0.147191	0.508	0.6121	6.953
Ethereum_2WeekReturn	0.086719	0.129136	0.672	0.5024	9.272
EthereumClassic_2WeekReturn	-0.113232	0.056013	-2.022	0.0441	4.565
Tether_2WeekReturn	3.584065	2.698965	1.328	0.1852	1.771
XRP_2WeekReturn	-0.097874	0.073994	-1.323	0.1869	4.758
Cardano_2WeekReturn	0.106163	0.083264	1.275	0.2033	6.216
Dogecoin_2WeekReturn	0.099158	0.025783	3.846	0.0001	3.071
Model Statistics					
Residual standard error: 14.26 on 302 degrees of freedom					
Multiple R-squared: 0.1294. Adjusted R-squared: 0.04296					
F-statistic: 1.497 on 30 and 302 DF. p-value: 0.05008					

Table 5.38: Selected Variables for Submodels on Cardano

Criterion	Variables Selected
BIC	Dogecoin_2WeekReturn
AIC	Dogecoin_1WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn, XAU_USD_1WeekReturn XAU_USD_2WeekReturn

Table 5.39: Average and Standard Deviation of Prediction Errors for Cardano Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	242.7052	98.60440
Submodel BIC	205.9436	53.51658
Shrinkage Estimator BIC	212.6747	65.22464
Positive Shrinkage Estimator BIC	212.6702	65.22562
Submodel AIC	211.7227	58.23939
Shrinkage Estimator AIC	218.0716	75.18130
Positive Shrinkage Estimator AIC	218.0283	75.21210
Submodel LASSO	241.3132	98.30585
Shrinkage Estimator LASSO	29706.3703	446855.47872
Positive Shrinkage Estimator LASSO	29706.3703	446855.47872
LASSO	241.6357	97.80146
ENET	240.7816	97.04326
ALASSO	241.1532	96.90917
SCAD	211.8651	61.58094
RIDGE	224.0466	68.00187

Table 5.40: Coefficients and Model Statistics for the Full Model of Dogecoin

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	3.456874	2.316299	1.492	0.1366	-
SP500_1WeekReturn	12.764033	10.618329	1.202	0.2303	166.637
DowJones_1WeekReturn	-6.698523	6.034964	-1.110	0.2679	55.868
NASDAQ_1WeekReturn	-10.566164	15.040709	-0.703	0.4829	434.489
Russell2000_1WeekReturn	0.066625	2.904864	0.023	0.9817	21.721
Nasdaq100_1WeekReturn	6.693339	12.792585	0.523	0.6012	320.096
DJOilGas_1WeekReturn	-1.038675	1.103378	-0.941	0.3473	5.460
DJGoldMining_1WeekReturn	-1.511545	1.086782	-1.391	0.1653	5.269
XAU_USD_1WeekReturn	4.137140	2.560172	1.616	0.1071	5.137
Bitcoin_1WeekReturn	-0.008662	0.646820	-0.013	0.9893	7.754
Ethereum_1WeekReturn	-0.822767	0.564591	-1.457	0.1461	9.874
EthereumClassic_1WeekReturn	-0.140828	0.224172	-0.628	0.5303	4.223
Tether_1WeekReturn	0.113066	9.534740	0.012	0.9905	1.882
XRP_1WeekReturn	1.016156	0.324334	3.133	0.0019	4.845
Cardano_1WeekReturn	0.267828	0.381947	0.701	0.4837	6.479
Dogecoin_1WeekReturn	-0.081812	0.092501	-0.884	0.3772	2.883
SP500_2WeekReturn	-7.807505	7.650807	-1.020	0.3083	155.321
DowJones_2WeekReturn	3.817333	4.370839	0.873	0.3832	52.082
NASDAQ_2WeekReturn	10.398019	11.057015	0.940	0.3478	436.782
Russell2000_2WeekReturn	-0.660483	2.254450	-0.293	0.7697	23.531
Nasdaq100_2WeekReturn	-5.994115	9.346209	-0.641	0.5218	316.615
DJOilGas_2WeekReturn	0.375555	0.768570	0.489	0.6255	5.535
DJGoldMining_2WeekReturn	0.389149	0.764230	0.509	0.6110	4.795
XAU_USD_2WeekReturn	-1.588734	1.769126	-0.898	0.3699	4.666
Bitcoin_2WeekReturn	-0.304748	0.411181	-0.741	0.4592	6.953
Ethereum_2WeekReturn	0.508022	0.360744	1.408	0.1601	9.272
EthereumClassic_2WeekReturn	0.138386	0.156474	0.884	0.3772	4.565
Tether_2WeekReturn	3.448886	7.539635	0.457	0.6477	1.771
XRP_2WeekReturn	-0.292825	0.206705	-1.417	0.1576	4.758
Cardano_2WeekReturn	-0.240767	0.232600	-1.035	0.3014	6.216
Dogecoin_2WeekReturn	0.040639	0.072026	0.564	0.5730	3.071
Model Statistics					
Residual standard error: 39.82 on 302 degrees of freedom					
Multiple R-squared: 0.1038. Adjusted R-squared: 0.01482					
F-statistic: 1.166 on 30 and 302 DF. p-value: 0.2568					

Table 5.41: Selected Variables for Submodels on Dogecoin

Criterion	Variables Selected
BIC	XRP_1WeekReturn
AIC	Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, Ethereum_1WeekReturn, XRP_1WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, XRP_2WeekReturn, XAU_USD_1WeekReturn
LASSO	SP500_1WeekReturn, DowJones_1WeekReturn, NASDAQ_1WeekReturn, Russell2000_1WeekReturn, Nasdaq100_1WeekReturn, DJOilGas_1WeekReturn, DJGoldMining_1WeekReturn, XAU_USD_1WeekReturn, Bitcoin_1WeekReturn, Ethereum_1WeekReturn, EthereumClassic_1WeekReturn, Tether_1WeekReturn, XRP_1WeekReturn, Cardano_1WeekReturn, Dogecoin_1WeekReturn, SP500_2WeekReturn, DowJones_2WeekReturn, NASDAQ_2WeekReturn, Russell2000_2WeekReturn, Nasdaq100_2WeekReturn, DJOilGas_2WeekReturn, DJGoldMining_2WeekReturn, XAU_USD_2WeekReturn, Bitcoin_2WeekReturn, Ethereum_2WeekReturn, EthereumClassic_2WeekReturn, Tether_2WeekReturn, XRP_2WeekReturn, Cardano_2WeekReturn, Dogecoin_2WeekReturn

Table 5.42: Average and Standard Deviation of Prediction Errors for Dogecoin Weekly Return

Model	**PE Average**	**PE s.d**
Full Model	1643.216	1483.775
Submodel BIC	1475.728	1587.647
Shrinkage Estimator BIC	1488.070	1498.006
Positive Shrinkage Estimator BIC	1487.943	1497.955
Submodel AIC	1466.038	1450.405
Shrinkage Estimator AIC	1504.730	1455.404
Positive Shrinkage Estimator AIC	1501.944	1455.410
Submodel LASSO	1643.216	1483.775
Shrinkage Estimator LASSO	-	-
Positive Shrinkage Estimator LASSO	-	-
LASSO	1636.639	1483.122
ENET	1634.026	1483.252
ALASSO	1634.479	1482.098
SCAD	1496.549	1552.910
RIDGE	1560.292	1481.069

5.4 Regression models with Principal Components

To address multicollinearity and reduce dimensionality among financial predictors, we adopt a methodology consistent with that presented in Chapter 4. Principal Component Analysis is applied separately to two groups of variables: stock indices and commodity indices. For each group, the first principal component is extracted to capture the dominant variance structure within its category. These components were then integrated into regression models alongside the weekly returns of major cryptocurrencies to forecast the weekly return of each target cryptocurrency. By summarizing complex market dynamics into a few orthogonal factors, the inclusion of principal components enhances model interpretability, mitigates overfitting and improves robustness.

5.4.1 Models for the Current-Week Returns

The PCA results for one-week returns of stock and commodity indices are summarized in Tables 5.43 and 5.44. In both groups, the first principal component captures a substantial proportion of the total variance, approximately 89.48% for stock indices and 60.62% for commodity indices. These results indicate that each group exhibits strong internal correlation, particularly the stock indices, which are dominated by a single underlying factor. These findings validate the use of PCA as an effective dimensionality reduction technique in financial modeling. By capturing the majority of variability in a few orthogonal components, PCA enables the construction of more parsimonious and interpretable regression models, while mitigating issues related to multicollinearity and overfitting.

Subsequently, we fit a linear regression model for each cryptocurrency to predict its current week's return. Each model incorporates the weekly returns of six other major

Table 5.43: Importance of components in PCA of stock indices 1-week returns

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**
Standard deviation	6.3455	1.8120	1.1615	0.2713	0.1609
Proportion of Variance	0.8948	0.0730	0.0300	0.0016	0.0006
Cumulative Proportion	0.8948	0.9678	0.9978	0.9994	1.0000

Table 5.44: Importance of components in PCA of Gold and Oil indices 1-week returns

	Comp.1	**Comp.2**	**Comp.3**
Standard deviation	5.2934	4.0884	1.2206
Proportion of Variance	0.6062	0.3616	0.0322
Cumulative Proportion	0.6062	0.9678	1.0000

cryptocurrencies along with the two principal components derived from stock and commodity indices. For example, the model for Bitcoin is specified as follows,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \varepsilon \tag{5.5}$$

Where

- Y: Weekly return of Bitcoin
- X_1: Stock_PC1 value, where Stock_PC1 is the first principal component derived from weekly returns of various stock indices: S&P 500, Dow Jones, NASDAQ, Nasdaq 100, Russell 2000. (Explains 89.48% of the total variability.)

- X_2: GoldOils_PC1 value, where GoldOils_PC1 is the first principal component derived from weekly returns of various commodity indices: DJ Oil & Gas, XAU/USD, DJ Gold Mining. (Explains 60.62% of the total variability.)
- X_3: Weekly return of Ethereum
- X_4: Weekly return of Ethereum Classic
- X_5: Weekly return of Tether
- X_6: Weekly return of XRP
- X_7: Weekly return of Cardano
- X_8: Weekly return of Dogecoin
- $\beta_0, \ldots, \beta_8$: Unknown regression parameters
- ε: Random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{334\times 1} = \mathbf{X}_{334\times 9}\beta_{9\times 1} + \varepsilon_{334\times 1}, \;\; Rank(\mathbf{X}) = 9 < 334 \tag{5.6}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin weekly returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 9$ predictors including the intercept, β is the vector of unknown regression parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

Table 5.45: Model Statistics for Bitcoin Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.2971	0.3121	0.952	0.3418	
Stock_PC1	-0.0834	0.0593	-1.406	0.1608	1.476
GoldOils_PC1	0.1396	0.0701	1.993	0.0471	1.433
Ethereum_1WeekReturn	0.5218	0.0411	12.690	$< 2 \times 10^{-16}$	2.592
EthereumClassic_1WeekReturn	-0.0119	0.0204	-0.582	0.5608	1.725
TetherUSDt_1WeekReturn	0.0816	0.9972	0.082	0.9348	1.018
XRP_1WeekReturn	0.0084	0.0284	0.296	0.7677	1.835
Cardano_1WeekReturn	0.1026	0.0335	3.063	0.0024	2.468
Dogecoin_1WeekReturn	0.0048	0.0087	0.546	0.5852	1.273
Model Statistics					
Residual standard error: 5.663 on 325 degrees of freedom					
Multiple R-squared: 0.6459					
Adjusted R-squared: 0.6371					
F-statistic: 74.09 on 8 and 325 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 38.2030, Standard deviation 26.4721					

As presented in Table 5.45, the regression model for Bitcoin's weekly return demonstrates strong explanatory power, with an R^2 of 0.6459 and a statistically significant overall fit (F-statistic = 74.09, $p < 2.2 \times 10^{-16}$). Compared to the R^2 of 0.6618 reported in Section 5.2, the incorporation of principal components slightly reduces the proportion of variance explained, reflecting a trade-off between dimensionality reduction and model fit. Among the predictors, Ethereum and Cardano weekly returns are statistically significant and positively associated with Bitcoin's performance, highlighting the strong interdependence among major cryptocurrencies. The first principal component of commodity indices (GoldOils_PC1) shows marginal significance (p-value = 0.0471), suggesting a modest influence from commodity markets. Other variables, including the principal components for stock indices and other cryptocurrency predictors do not show

statistically significant effects. These results emphasize that Bitcoin's short-term weekly movements are primarily driven by internal crypto-market dynamics, while broader macroeconomic indicators play a secondary role. All variance inflation factors are below 3, confirming that multicollinearity has been effectively mitigated through PCA. The bootstrap-based prediction errors has average of 38.2030, which is slightly higher than 37.97662 reported for the full model estimator in Section 5.2 (Table 5.3), suggesting a minor reduction in predictive accuracy.

Table 5.46: Model Statistics for Ethereum Weekly Return Model with Principal Components

Variable	**Estimate**	**Std. Error**	**t value**	**P value**	**VIF**
(Intercept)	0.1227	0.3447	0.356	0.7221	
Stock_PC1	0.1759	0.0649	2.710	0.0071	1.452
GoldOils_PC1	-0.0392	0.0778	-0.504	0.6148	1.450
Bitcoin_1WeekReturn	0.6350	0.0500	12.690	$< 2 \times 10^{-16}$	1.888
EthereumClassic_1WeekReturn	0.1095	0.0216	5.060	7.02×10^{-7}	1.601
TetherUSDt_1WeekReturn	-1.0894	1.0985	-0.992	0.3221	1.015
XRP_1WeekReturn	0.0829	0.0310	2.678	0.0078	1.796
Cardano_1WeekReturn	0.1876	0.0360	5.210	3.36×10^{-7}	2.344
Dogecoin_1WeekReturn	0.0022	0.0096	0.230	0.8180	1.274
Model Statistics					
Residual standard error: 6.247 on 325 degrees of freedom					
Multiple R-squared: 0.7420					
Adjusted R-squared: 0.7357					
F-statistic: 116.9 on 8 and 325 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 39.1701, Standard deviation 10.7789					

For Ethereum's weekly return, the regression model achieves an R^2 of 0.7420 and a highly significant overall fit (F-statistic = 116.9, $p < 2.2 \times 10^{-16}$), indicating strong

explanatory power, as shown in Table 5.46. Compared to the R^2 of 0.7431 reported in the previous section, the use of principal components yields a comparable level of explained variance. The weekly returns of Bitcoin, Ethereum Classic, XRP, and Cardano are statistically significant and positively associated with Ethereum's performance, reflecting the high degree of interconnectedness within the cryptocurrency market. Additionally, the stock market principal component (Stock_PC1) is statistically significant ($p = 0.0071$), suggesting that equity market trends may exert a modest influence on Ethereum's weekly movements. These results emphasize that Ethereum's short-term weekly behavior is primarily shaped by internal crypto-market dynamics, with limited but notable contributions from broader financial indicators. All VIF values are below 2.5, confirming that multicollinearity has been effectively mitigated through the use of PCA. The bootstrap-based prediction error has an average of 39.1701, which is lower than the 41.0188 reported in the previous section without principal components, indicating improved predictive performance and reduced variability.

The regression results for Ethereum Classic's weekly return using principal components is presented in Table 5.47. The model achieves an R^2 of 0.4208 with a statistically significant overall fit (F-statistic = 29.52, $p < 2.2 \times 10^{-16}$). Compared to the R^2 of 0.4467 from the full model without principal components in previous section, the explanatory power is slightly reduced, reflecting a trade-off between dimensionality reduction and model fit. Among the predictors, the weekly returns of Ethereum, Cardano, and Dogecoin are statistically significant and positively associated with Ethereum Classic's performance, highlighting strong intra-market relationships among cryptocurrencies. The commodity principal component (GoldOils_PC1) is marginally significant ($p = 0.0682$), suggesting a modest influence from commodity markets. Other predictors, including Bitcoin, XRP, Tether, and the stock principal component, do not

Table 5.47: Model Statistics for Ethereum Classic Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.0298	0.8507	0.035	0.9721	
Stock_PC1	-0.0369	0.1620	-0.228	0.8200	1.485
GoldOils_PC1	0.3493	0.1910	1.829	0.0682	1.436
Bitcoin_1WeekReturn	-0.0879	0.1509	-0.582	0.5608	2.821
Ethereum_1WeekReturn	0.6668	0.1318	5.060	7.02×10^{-7}	3.594
TetherUSDt_1WeekReturn	1.8952	2.7126	0.699	0.4853	1.016
XRP_1WeekReturn	0.0140	0.0773	0.181	0.8568	1.835
Cardano_1WeekReturn	0.1937	0.0919	2.108	0.0358	2.505
Dogecoin_1WeekReturn	0.1267	0.0227	5.575	5.20×10^{-8}	1.163
Model Statistics					
Residual standard error: 15.41 on 325 degrees of freedom					
Multiple R-squared: 0.4208					
Adjusted R-squared: 0.4066					
F-statistic: 29.52 on 8 and 325 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 300.1929, Standard deviation 372.5872					

exhibit statistically significant effects. These results suggest that Ethereum Classic's short-term weekly movements are primarily driven by select cryptocurrency signals, with limited contribution from broader macroeconomic indicators. While the VIF for Ethereum is relatively high (3.594), all other VIFs remain below 3, indicating that PCA has helped mitigate multicollinearity compared to the previous model. The bootstrap-based prediction error has an average of 300.1929, slightly higher than the 296.1885 reported for the full model from section 5.2, indicating a minor reduction in predictive accuracy but improved model stability and interpretability.

As presented in Table 5.48, the regression model for Tether's weekly return exhibits very limited explanatory power, with an R^2 of 0.0177. The overall model fit is not statistically significant (F-statistic = 0.7309, $p = 0.6642$), indicating that the included

Table 5.48: Model Statistics for Tether Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.0011	0.0174	-0.063	0.9500	
Stock_PC1	0.0055	0.0033	1.676	0.0947	1.472
GoldOils_PC1	-0.0008	0.0039	-0.192	0.8480	1.451
Bitcoin_1WeekReturn	0.0003	0.0031	0.082	0.9348	2.824
Ethereum_1WeekReturn	-0.0028	0.0028	-0.992	0.3221	3.865
EthereumClassic_1WeekReturn	0.0008	0.0011	0.699	0.4853	1.724
XRP_1WeekReturn	-0.0014	0.0016	-0.899	0.3695	1.831
Cardano_1WeekReturn	0.0016	0.0019	0.823	0.4112	2.534
Dogecoin_1WeekReturn	0.0004	0.0005	0.744	0.4572	1.272
Model Statistics					
Residual standard error: 0.315 on 325 degrees of freedom					
Multiple R-squared: 0.0177					
Adjusted R-squared: -0.0065					
F-statistic: 0.7309 on 8 and 325 DF					
p-value: 0.6642					
Bootstrap sample prediction errors: Average 0.1021, Standard deviation 0.0314					

predictors do not meaningfully explain Tether's weekly movements. None of the predictors are statistically significant, and all estimated coefficients are small in magnitude. These results are consistent with the nature of Tether as a stablecoin, which is designed to maintain a fixed value and thus exhibits minimal volatility. Compared to the model without principal components presented in the previous section ($R^2 = 0.0642$, $p = 0.08806$), the inclusion of PCs slightly reduces the explained variance but refines multicollinearity, as evidenced by lower VIFs (all below 4). The bootstrap-based prediction error has an average of 0.1021, which is marginally lower than the 0.1039 reported for the full model without PCs, suggesting a slight improvement in predictive stability despite the overall weak model performance.

Table 5.49: Model Statistics for XRP Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.2478	0.6107	-0.406	0.6852	
Stock_PC1	0.1125	0.1161	0.969	0.3332	1.480
GoldOils_PC1	-0.0483	0.1378	-0.350	0.7264	1.450
Bitcoin_1WeekReturn	0.0320	0.1084	0.296	0.7677	2.823
Ethereum_1WeekReturn	0.2603	0.0972	2.678	0.0078	3.793
EthereumClassic_1WeekReturn	0.0072	0.0398	0.181	0.8568	1.726
TetherUSDt_1WeekReturn	-1.7495	1.9468	-0.899	0.3695	1.015
Cardano_1WeekReturn	0.4304	0.0620	6.944	2.09×10^{-11}	2.211
Dogecoin_1WeekReturn	0.0434	0.0169	2.565	0.0108	1.248
Model Statistics					
Residual standard error: 11.07 on 325 degrees of freedom					
Multiple R-squared: 0.4551					
Adjusted R-squared: 0.4417					
F-statistic: 33.94 on 8 and 325 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 130.8568, Standard deviation 59.9878					

The Table 5.49 presents the regression model statistics for predicting XRP's weekly returns. The model demonstrates moderate explanatory power, with an R^2 of 0.4551 and a statistically significant overall fit (F-statistic = 33.94, $p < 2.2 \times 10^{-16}$), indicating that the selected predictors contribute meaningfully to explaining XRP's weekly movements. Among the predictors, Cardano's weekly return shows the strongest positive association with XRP, followed by Ethereum and Dogecoin, all of which are statistically significant. Other variables, including Bitcoin, Ethereum Classic, Tether, and the principal components for stock and commodity indices, do not exhibit significant effects. These results suggest that XRP's short-term behavior is primarily influenced by internal crypto-market dynamics rather than broader financial indicators. All VIFs are below 4, indicating acceptable levels of multicollinearity. The bootstrap-based prediction

error has an average of 130.8568, which is slightly higher than the 130.6080 reported in the previous section without principal components.

Table 5.50: Model Statistics for Cardano Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.1367	0.5101	-0.268	0.7889	
Stock_PC1	0.0314	0.0971	0.324	0.7463	1.484
GoldOils_PC1	-0.0109	0.1151	-0.095	0.9244	1.451
Bitcoin_1WeekReturn	0.2734	0.0893	3.063	0.0024	2.745
Ethereum_1WeekReturn	0.4108	0.0789	5.210	3.36×10^{-7}	3.578
EthereumClassic_1WeekReturn	0.0696	0.0330	2.108	0.0358	1.703
TetherUSDt_1WeekReturn	1.3380	1.6262	0.823	0.4112	1.016
XRP_1WeekReturn	0.3002	0.0432	6.944	2.09×10^{-11}	1.598
Dogecoin_1WeekReturn	-0.0018	0.0143	-0.124	0.9016	1.274
Model Statistics					
Residual standard error: 9.244 on 325 degrees of freedom					
Multiple R-squared: 0.6062					
Adjusted R-squared: 0.5965					
F-statistic: 62.54 on 8 and 325 DF					
p-value: $< 2.2 \times 10^{-16}$					
Bootstrap sample prediction errors: Average 94.1384, Standard deviation 48.6265					

As presented in Table 5.50 , the regression model for Cardano's weekly return demonstrates solid explanatory power, with an R^2 of 0.6062. The overall model fit is statistically significant (F-statistic = 62.54, $p < 2.2 \times 10^{-16}$), indicating that the selected predictors meaningfully explain Cardano's weekly movements. Among the predictors, Ethereum, XRP and Bitcoin show strong positive associations with Cardano, highlighting the interconnectedness within the crypto market. Ethereum Classic also contributes positively ($p = 0.0358$), though to a lesser extent. Other cryptocurrencies and the principal components for stock and commodity indices are not statistically significant,

implying limited influence from aggregated macroeconomic indicators. All VIFs are below 4, indicating that multicollinearity has been effectively mitigated through PCA. The bootstrap-based prediction error has an average of 94.1384, which is notably lower than the 100.4276 reported in the previous section without principal components, suggesting improved predictive accuracy and model stability.

Table 5.51: Model Statistics for Dogecoin Weekly Return Model with Principal Components

Variable	**Estimate**	**Std. Error**	**t value**	**P value**	**VIF**
(Intercept)	2.3474	1.9790	1.186	0.2364	
Stock_PC1	-0.4722	0.3767	-1.254	0.2109	1.478
GoldOils_PC1	-0.1103	0.4474	-0.246	0.8055	1.451
Bitcoin_1WeekReturn	0.1922	0.3519	0.546	0.5852	2.821
Ethereum_1WeekReturn	0.0735	0.3191	0.230	0.8180	3.876
EthereumClassic_1WeekReturn	0.6888	0.1235	5.575	5.20×10^{-8}	1.576
TetherUSDt_1WeekReturn	4.7065	6.3231	0.744	0.4572	1.016
XRP_1WeekReturn	0.4574	0.1783	2.565	0.0108	1.799
Cardano_1WeekReturn	-0.0267	0.2156	-0.124	0.9016	2.539
Model Statistics					
Residual standard error: 35.94 on 325 degrees of freedom					
Multiple R-squared: 0.2149					
Adjusted R-squared: 0.1956					
F-statistic: 11.12 on 8 and 325 DF					
p-value: 6.598×10^{-14}					
Bootstrap sample prediction errors: Average 1552.5178, Standard deviation 1599.8010					

The regression model for Dogecoin's weekly return using principal components yields a relatively low explanatory power, with an R^2 of 0.2149, as shown in Table 5.51. The overall model fit is statistically significant (F-statistic = 11.12, $p = 6.598 \times 10^{-14}$), indicating that the predictors contribute meaningfully, albeit modestly, to explaining

Dogecoin's weekly movements. Among the predictors, Ethereum Classic and XRP weekly returns are statistically significant, both exhibiting positive associations with Dogecoin's performance, suggesting some inter-cryptocurrency influence from these coins. Other variables do not show statistically significant effects. These findings imply that Dogecoin's weekly return behavior is not effectively captured by aggregated macroeconomic indicators and is only marginally explained by select cryptocurrency-specific variables. While the VIF for Ethereum is relatively high (3.876), all other VIFs remain below 3, indicating that PCA has helped reduce multicollinearity compared to the previous model. The bootstrap-based prediction error has an average of 1552.5178, which is higher than the 1386.007 reported in Section 5.2 for the full model without principal components, suggesting a slight decline in predictive accuracy despite improved model interpretability.

5.4.2 Models for the Next-One-Week Returns

To forecast the one-week-ahead returns of each cryptocurrency, we construct a regression framework that incorporate lagged returns from the preceding one-week and two-week periods. For each lag period, Principal Component Analysis is independently applied to two distinct groups of financial indices: stock indices and commodity indices. The first principal component (PC1) from each PCA is extracted to capture the dominant variance structure within each financial group and time horizon. This process yields four first principal components in total, namely two for each financial group across the two lag periods. These components are then used as predictors in regression models, alongside the one-week and two-week lagged returns of all cryptocurrencies. By summarizing complex interdependencies into orthogonal components, this approach effectively mitigates multicollinearity and enhances model interpretability.

Table 5.52: Importance of components in PCA of stock indices 2-week returns

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**
Standard deviation	8.5699	2.4116	1.5722	0.3737	0.2183
Proportion of Variance	0.8965	0.0710	0.0302	0.0017	0.0006
Cumulative Proportion	0.8965	0.9675	0.9977	0.9994	1.0000

Table 5.53: Importance of components in PCA of Gold and Oil indices 2-week returns

	Comp.1	**Comp.2**	**Comp.3**
Standard deviation	7.2736	5.8921	1.7530
Proportion of Variance	0.5833	0.3828	0.0339
Cumulative Proportion	0.5833	0.9661	1.0000

The PCA results for the 1-week lagged returns of each financial group are presented in the preceding section (Tables 5.43 and 5.44). The corresponding PCA results for the 2-week lagged returns of stock and commodity indices are summarized in Tables 5.52 and 5.53, respectively. Across all financial groups and time horizons, the first principal component consistently captures the majority of the variance. For stock indices, PC1 accounts for approximately 90% of the variance, reflecting strong co-movement among major equity indices. For commodity indices, PC1 captures over 58% of the variance, suggesting a moderate-to-strong correlation between gold and oil indices. These results validate the use of PCA for dimensionality reduction, as the extracted components effectively summarize complex interdependencies and serve as robust predictors in the regression models for cryptocurrency returns.

Subsequently, we estimate a linear regression model for each cryptocurrency's next-week return. The predictors include the first principal components extracted from each PCA corresponding to the 1-week and 2-week lagged return horizons of stock indices and commodity indices, along with the lagged 1-week and 2-week returns of all cryptocurrencies. For example, the model for predicting Bitcoin's next-week return is expressed as:

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 \\ &+ \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} + \beta_{15} X_{15} \\ &+ \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \varepsilon \end{aligned} \tag{5.7}$$

Where

- Y: Next week's return of Bitcoin
- X_1 to X_9: Returns over the past one week
  - X_1: Stock_PC1 for past one week value, where Stock_PC1 is the first principal component derived from returns of various stock indices from past one week: S&P 500, Dow Jones, NASDAQ, Nasdaq 100, Russell 2000. (Explains 89.48% of the total variability.)
  - X_2: GoldOils_PC1 for past one week value, where GoldOils_PC1 is the first principal component derived from returns of various commodity indices from past one week: DJ Oil & Gas, XAU/USD, DJ Gold Mining. (Explains 60.62% of the total variability.)
  - X_3: Bitcoin (past one week return)
  - X_4: Ethereum

  - X_5: Ethereum Classic
  - X_6: Tether
  - X_7: XRP
  - X_8: Cardano
  - X_9: Dogecoin
- X_{10} to X_{18}: Cumulative returns over the two weeks prior (same structure as above)
- $\beta_0, \ldots, \beta_{18}$: Unknown regression parameters
- ε: Random error term

We assume errors to be randomly and independently distributed as normal with $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$. In matrix notation, the model is expressed as

$$\mathbf{Y}_{333\times 1} = \mathbf{X}_{333\times 18}\beta_{18\times 1} + \varepsilon_{333\times 1}, \;\; Rank(\mathbf{X}) = 18 < 333 \tag{5.8}$$

where $\mathbf{Y}$ denotes the response vector of Bitcoin next week's returns, $\mathbf{X}$ is the design matrix of predictors containing $p = 18$ predictors including the intercept, β is the vector of unknown regression parameters, and ε is the vector of random errors. We assume errors to be randomly and independently distributed as normal with E(ε) = 0, and $E(\varepsilon\varepsilon') = \sigma^2 I_n$, where $\sigma^2 > 0$.

The regression model statistics for predicting one-week-ahead returns of various cryptocurrencies are presented in Table 5.54 - 5.60. The predictive performance of these models is generally limited, with low R^2 values across all assets ranging from 0.0486 for Ethereum to 0.1520 for Tether, indicating that the models explain only a small portion of the variability in next-week returns. Compared to the full models without principal

Table 5.54: Model Statistics for Bitcoin Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.8786	0.5233	1.679	0.0941	
Stock_PC1_1Week	0.0312	0.1395	0.223	0.8233	2.978
GoldOil_PC1_1Week	-0.1711	0.1675	-1.021	0.3079	2.991
Bitcoin_1WeekReturn	0.1492	0.1456	1.025	0.3063	7.093
Ethereum_1WeekReturn	-0.1550	0.1306	-1.187	0.2363	9.542
EthereumClassic_1WeekReturn	0.0242	0.0518	0.468	0.6401	4.068
TetherUSDt_1WeekReturn	-0.3001	2.1331	-0.141	0.8882	1.701
XRP_1WeekReturn	-0.0123	0.0744	-0.166	0.8684	4.607
Cardano_1WeekReturn	0.0465	0.0882	0.527	0.5987	6.243
Dogecoin_1WeekReturn	-0.0357	0.0214	-1.666	0.0967	2.788
Stock_PC1_2Week	0.0987	0.1029	0.960	0.3378	2.953
GoldOil_PC1_2Week	0.0033	0.1207	0.027	0.9783	2.922
Bitcoin_2WeekReturn	0.0372	0.0928	0.401	0.6890	6.393
Ethereum_2WeekReturn	0.0234	0.0835	0.281	0.7791	8.970
EthereumClassic_2WeekReturn	-0.0680	0.0358	-1.900	0.0584	4.318
TetherUSDt_2WeekReturn	1.5414	1.7083	0.902	0.3676	1.642
XRP_2WeekReturn	-0.0119	0.0476	-0.250	0.8029	4.566
Cardano_2WeekReturn	0.0068	0.0537	0.126	0.8995	5.980
Dogecoin_2WeekReturn	0.0380	0.0167	2.280	0.0233	2.966
Model Statistics					
Residual standard error: 9.371 on 314 degrees of freedom					
Multiple R-squared: 0.0623					
Adjusted R-squared: 0.0085					
F-statistic: 1.158 on 18 and 314 DF					
p-value: 0.2954					
Bootstrap sample prediction errors: Average 95.3643, Standard deviation 26.7459					

components presented in Section 5.3, the proportion of explained variance is generally lower, suggesting that the inclusion of PCA-based macroeconomic features does not enhance predictive power. Most models exhibit signs of no or marginal statistical significance, with the exception of Tether and Cardano, whose models show stronger significance.

Among the predictors, only a few lagged cryptocurrency returns, such as those of Bitcoin, Ethereum, Tether, and Dogecoin, are statistically significant, and their effects vary inconsistently across models. Principal components derived from stock and commodity indices generally lack meaningful predictive influence in most cases. However, their inclusion substantially reduces multicollinearity, as evidenced by lower Variance Inflation Factor values. Prediction errors estimated via bootstrap sampling remain high, particularly for volatile assets like Dogecoin and Ethereum Classic. Compared to the full models in Section 5.3, average prediction errors are slightly larger, except for XRP, Cardano, and Dogecoin. The inclusion of principal components results in slightly higher average prediction errors for most cryptocurrencies. Specifically, Bitcoin, Ethereum, Ethereum Classic, and Tether exhibit increased error magnitudes, while XRP, Cardano, and Dogecoin show modest improvements in predictive accuracy. This mixed outcome suggests that while principal components help reduce multicollinearity, they do not consistently enhance predictive performance across all assets.

These findings suggest that next-week return prediction is substantially more difficult than current-week modeling. The temporal lag appears to weaken the strong intra-cryptocurrency relationships observed in shorter horizons. Additionally, the volatility and noise inherent in crypto markets likely obscure any signal that could be captured by lagged financial indicators or macroeconomic trends.

5.5 Concluding Remarks

This chapter extended the predictive modeling framework to the weekly return horizon, offering a broader temporal perspective on cryptocurrency behavior. By incorporating both current and lagged weekly returns of cryptocurrencies and financial indices, the analysis aimed to capture short-term dependencies and autocorrelations that

Table 5.55: Model Statistics for Ethereum Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.8316	0.6815	1.220	0.223	
Stock_PC1_1Week	0.0527	0.1817	0.290	0.772	2.978
GoldOil_PC1_1Week	-0.0957	0.2182	-0.439	0.661	2.991
Bitcoin_1WeekReturn	0.0034	0.1896	0.018	0.986	7.093
Ethereum_1WeekReturn	-0.1099	0.1701	-0.646	0.519	9.542
EthereumClassic_1WeekReturn	0.1143	0.0674	1.695	0.091	4.068
TetherUSDt_1WeekReturn	-0.5566	2.7783	-0.200	0.841	1.701
XRP_1WeekReturn	-0.0348	0.0969	-0.359	0.720	4.607
Cardano_1WeekReturn	0.0186	0.1149	0.161	0.872	6.243
Dogecoin_1WeekReturn	0.0009	0.0279	0.033	0.974	2.788
Stock_PC1_2Week	0.0225	0.1340	0.168	0.867	2.953
GoldOil_PC1_2Week	-0.0131	0.1571	-0.083	0.934	2.922
Bitcoin_2WeekReturn	0.0889	0.1208	0.736	0.463	6.393
Ethereum_2WeekReturn	0.0765	0.1088	0.703	0.483	8.970
EthereumClassic_2WeekReturn	-0.0962	0.0466	-2.062	0.040	4.318
TetherUSDt_2WeekReturn	3.6236	2.2250	1.629	0.104	1.642
XRP_2WeekReturn	-0.0130	0.0621	-0.209	0.835	4.566
Cardano_2WeekReturn	0.0181	0.0699	0.259	0.795	5.980
Dogecoin_2WeekReturn	0.0206	0.0217	0.951	0.342	2.966
Model Statistics					
Residual standard error: 12.21 on 314 degrees of freedom					
Multiple R-squared: 0.0486					
Adjusted R-squared: -0.0059					
F-statistic: 0.8917 on 18 and 314 DF					
p-value: 0.5891					
Bootstrap sample prediction errors: Average 164.0266, Standard deviation 42.3204					

are often overlooked in daily models. Compared to daily return models, weekly models for future returns demonstrated improved explanatory power, particularly for assets like Bitcoin, Ethereum, and Cardano.

The results revealed that inter-cryptocurrency relationships, particularly among Ethereum, Bitcoin, Cardano, and Ethereum Classic, consistently emerged as strong

Table 5.56: Model Statistics for Ethereum Classic Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	1.1965	1.0916	1.096	0.2739	
Stock_PC1_1Week	0.4310	0.2910	1.481	0.1395	2.978
GoldOil_PC1_1Week	-0.5649	0.3495	-1.616	0.1070	2.991
Bitcoin_1WeekReturn	0.8440	0.3037	2.779	0.0058	7.093
Ethereum_1WeekReturn	-1.0006	0.2725	-3.672	0.0003	9.542
EthereumClassic_1WeekReturn	0.1405	0.1080	1.301	0.1942	4.068
TetherUSDt_1WeekReturn	-0.1381	4.4501	-0.031	0.9753	1.701
XRP_1WeekReturn	0.3734	0.1553	2.405	0.0168	4.607
Cardano_1WeekReturn	0.1232	0.1841	0.669	0.5037	6.243
Dogecoin_1WeekReturn	-0.0182	0.0447	-0.408	0.6836	2.788
Stock_PC1_2Week	-0.0951	0.2146	-0.443	0.6581	2.953
GoldOil_PC1_2Week	0.0235	0.2517	0.094	0.9256	2.922
Bitcoin_2WeekReturn	-0.4999	0.1936	-2.583	0.0103	6.393
Ethereum_2WeekReturn	0.6518	0.1742	3.742	0.0002	8.970
EthereumClassic_2WeekReturn	-0.1098	0.0747	-1.469	0.1427	4.318
TetherUSDt_2WeekReturn	1.3305	3.5638	0.373	0.7091	1.642
XRP_2WeekReturn	-0.1138	0.0994	-1.144	0.2533	4.566
Cardano_2WeekReturn	-0.0956	0.1120	-0.854	0.3939	5.980
Dogecoin_2WeekReturn	0.0186	0.0348	0.534	0.5936	2.966
Model Statistics					
Residual standard error: 19.55 on 314 degrees of freedom					
Multiple R-squared: 0.0992					
Adjusted R-squared: 0.0476					
F-statistic: 1.921 on 18 and 314 DF					
p-value: 0.014					
Bootstrap sample prediction errors: Average 407.8381, Standard deviation 276.1648					

predictors of weekly returns. In contrast, traditional financial indices such as stock and commodity benchmarks exhibited limited and inconsistent influence, underscoring the relative autonomy of the crypto market from broader macroeconomic trends. Submodel selection techniques and penalized regression methods proved effective in improving predictive accuracy and model stability. Notably, AIC-selected submodels offered

Table 5.57: Model Statistics for Tether Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	-0.0021	0.0166	-0.127	0.8992	
Stock_PC1_1Week	-0.0003	0.0044	-0.059	0.9533	2.978
GoldOil_PC1_1Week	0.0003	0.0053	0.053	0.9577	2.991
Bitcoin_1WeekReturn	-0.0022	0.0046	-0.484	0.6287	7.093
Ethereum_1WeekReturn	0.0003	0.0041	0.064	0.9492	9.542
EthereumClassic_1WeekReturn	0.0004	0.0016	0.228	0.8197	4.068
TetherUSDt_1WeekReturn	-0.1368	0.0678	-2.018	0.0444	1.701
XRP_1WeekReturn	0.0016	0.0024	0.671	0.5025	4.607
Cardano_1WeekReturn	0.0043	0.0028	1.537	0.1254	6.243
Dogecoin_1WeekReturn	-0.0003	0.0007	-0.487	0.6263	2.788
Stock_PC1_2Week	-0.0063	0.0033	-1.935	0.0539	2.953
GoldOil_PC1_2Week	-0.0015	0.0038	-0.381	0.7035	2.922
Bitcoin_2WeekReturn	-0.0020	0.0029	-0.676	0.4996	6.393
Ethereum_2WeekReturn	0.0035	0.0027	1.307	0.1923	8.970
EthereumClassic_2WeekReturn	-0.0006	0.0011	-0.484	0.6285	4.318
TetherUSDt_2WeekReturn	-0.1500	0.0543	-2.763	0.0061	1.642
XRP_2WeekReturn	-0.0004	0.0015	-0.266	0.7907	4.566
Cardano_2WeekReturn	-0.0022	0.0017	-1.315	0.1895	5.980
Dogecoin_2WeekReturn	0.0001	0.0005	0.258	0.7964	2.966
Model Statistics					
Residual standard error: 0.2977 on 314 degrees of freedom					
Multiple R-squared: 0.1520					
Adjusted R-squared: 0.1034					
F-statistic: 3.126 on 18 and 314 DF					
p-value: 2.57×10^{-5}					
Bootstrap sample prediction errors: Average 0.0981, Standard deviation 0.0340					

superior accuracy in most cases, penalized methods such as Ridge and SCAD offered moderate improvements across cryptocurrencies. Next-week return prediction proved substantially more challenging than current-week modeling. Temporal lags weakened intra-cryptocurrency relationships, and the inherent volatility of crypto markets obscured signals from lagged financial indicators. Prediction errors were higher, and model

Table 5.58: Model Statistics for XRP Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.4453	0.8280	0.538	0.5911	
Stock_PC1_1Week	-0.0630	0.2207	-0.285	0.7756	2.978
GoldOil_PC1_1Week	-0.0734	0.2651	-0.277	0.7820	2.991
Bitcoin_1WeekReturn	-0.0078	0.2303	-0.034	0.9732	7.093
Ethereum_1WeekReturn	-0.0912	0.2067	-0.441	0.6592	9.542
EthereumClassic_1WeekReturn	-0.0187	0.0819	-0.229	0.8193	4.068
TetherUSDt_1WeekReturn	-3.3634	3.3753	-0.996	0.3198	1.701
XRP_1WeekReturn	0.1192	0.1178	1.012	0.3123	4.607
Cardano_1WeekReturn	0.0801	0.1396	0.574	0.5666	6.243
Dogecoin_1WeekReturn	-0.0260	0.0339	-0.767	0.4438	2.788
Stock_PC1_2Week	0.0556	0.1628	0.342	0.7329	2.953
GoldOil_PC1_2Week	-0.0003	0.1909	-0.001	0.9989	2.922
Bitcoin_2WeekReturn	0.0352	0.1468	0.240	0.8105	6.393
Ethereum_2WeekReturn	0.0971	0.1321	0.735	0.4627	8.970
EthereumClassic_2WeekReturn	-0.0452	0.0567	-0.797	0.4258	4.318
TetherUSDt_2WeekReturn	3.9427	2.7031	1.459	0.1457	1.642
XRP_2WeekReturn	-0.0196	0.0754	-0.260	0.7953	4.566
Cardano_2WeekReturn	-0.1184	0.0850	-1.394	0.1642	5.980
Dogecoin_2WeekReturn	0.0592	0.0264	2.247	0.0253	2.966
Model Statistics					
Residual standard error: 14.83 on 314 degrees of freedom					
Multiple R-squared: 0.0487					
Adjusted R-squared: -0.0058					
F-statistic: 0.8936 on 18 and 314 DF					
p-value: 0.5868					
Bootstrap sample prediction errors: Average 249.2468, Standard deviation 104.9468					

significance diminished across most assets, with the exception of Tether and Cardano, which retained moderate predictive strength.

Principal Component Analysis was applied to reduce dimensionality and mitigate multicollinearity among predictors. While PCA-based models effectively mitigated multicollinearity, they did not consistently enhance predictive performance. In most cases,

Table 5.59: Model Statistics for Cardano Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	0.2234	0.7862	0.284	0.7765	
Stock_PC1_1Week	0.3650	0.2096	1.741	0.0826	2.978
GoldOil_PC1_1Week	-0.4913	0.2517	-1.952	0.0518	2.991
Bitcoin_1WeekReturn	0.0907	0.2187	0.414	0.6788	7.093
Ethereum_1WeekReturn	-0.2145	0.1962	-1.093	0.2752	9.542
EthereumClassic_1WeekReturn	0.1374	0.0778	1.766	0.0783	4.068
TetherUSDt_1WeekReturn	-3.0847	3.2052	-0.962	0.3366	1.701
XRP_1WeekReturn	0.0016	0.1118	0.014	0.9885	4.607
Cardano_1WeekReturn	0.0237	0.1326	0.179	0.8584	6.243
Dogecoin_1WeekReturn	-0.0518	0.0322	-1.610	0.1084	2.788
Stock_PC1_2Week	-0.1385	0.1546	-0.896	0.3709	2.953
GoldOil_PC1_2Week	0.1967	0.1813	1.085	0.2788	2.922
Bitcoin_2WeekReturn	0.0650	0.1394	0.466	0.6414	6.393
Ethereum_2WeekReturn	0.0923	0.1255	0.736	0.4626	8.970
EthereumClassic_2WeekReturn	-0.1090	0.0538	-2.025	0.0437	4.318
TetherUSDt_2WeekReturn	3.7006	2.5668	1.442	0.1504	1.642
XRP_2WeekReturn	-0.0951	0.0716	-1.329	0.1849	4.566
Cardano_2WeekReturn	0.0895	0.0807	1.109	0.2683	5.980
Dogecoin_2WeekReturn	0.0984	0.0250	3.932	0.0001	2.966
Model Statistics					
Residual standard error: 14.08 on 314 degrees of freedom					
Multiple R-squared: 0.1169					
Adjusted R-squared: 0.0663					
F-statistic: 2.309 on 18 and 314 DF					
p-value: 0.002076					
Bootstrap sample prediction errors: Average 225.4692, Standard deviation 79.7374					

models using raw financial indicators outperformed those using principal components, suggesting that direct relationships among variables may be more informative than aggregated latent structures. The findings underscore the importance of targeted variable selection, dimensionality reduction, and robust estimation strategies in modeling

Table 5.60: Model Statistics for Dogecoin Future Weekly Return Model with Principal Components

Variable	Estimate	Std. Error	t value	P value	VIF
(Intercept)	4.1145	2.2066	1.865	0.0632	
Stock_PC1_1Week	0.7214	0.5882	1.227	0.2209	2.978
GoldOil_PC1_1Week	-0.6746	0.7064	-0.955	0.3404	2.991
Bitcoin_1WeekReturn	0.1133	0.6139	0.184	0.8538	7.093
Ethereum_1WeekReturn	-0.9811	0.5508	-1.781	0.0758	9.542
EthereumClassic_1WeekReturn	-0.0729	0.2183	-0.334	0.7387	4.068
TetherUSDt_1WeekReturn	0.5954	8.9954	0.066	0.9473	1.701
XRP_1WeekReturn	0.9746	0.3139	3.105	0.0021	4.607
Cardano_1WeekReturn	0.2755	0.3721	0.741	0.4595	6.243
Dogecoin_1WeekReturn	-0.0755	0.0903	-0.836	0.4038	2.788
Stock_PC1_2Week	0.1967	0.4338	0.453	0.6506	2.953
GoldOil_PC1_2Week	-0.1158	0.5088	-0.228	0.8202	2.922
Bitcoin_2WeekReturn	-0.3543	0.3913	-0.905	0.3659	6.393
Ethereum_2WeekReturn	0.5604	0.3521	1.591	0.1125	8.970
EthereumClassic_2WeekReturn	0.1145	0.1510	0.758	0.4490	4.318
TetherUSDt_2WeekReturn	3.6671	7.2039	0.509	0.6111	1.642
XRP_2WeekReturn	-0.2611	0.2009	-1.300	0.1947	4.566
Cardano_2WeekReturn	-0.2417	0.2264	-1.067	0.2866	5.980
Dogecoin_2WeekReturn	0.0230	0.0702	0.327	0.7439	2.966
Model Statistics					
Residual standard error: 39.52 on 314 degrees of freedom					
Multiple R-squared: 0.0825					
Adjusted R-squared: 0.0299					
F-statistic: 1.568 on 18 and 314 DF					
p-value: 0.0668					
Bootstrap sample prediction errors: Average 1551.4780, Standard deviation 1507.8430					

cryptocurrency returns. These insights lay the groundwork for future research into portfolio optimization and algorithmic trading strategies within the crypto domain.

CHAPTER SIX

AN EXPLORATION OF PAIR-TRADING FOR CRYPTOCURRENCIES

6.1 Introduction

Pair trading is a market-neutral strategy that exploits relative price movements between two historically correlated assets, aiming to profit from temporary deviations in their relative prices. This chapter explores the viability of pair-trading strategies among major cryptocurrencies by leveraging the cointegration analysis and spread-based trading rules.

Fil and Kristoufek (2020) have examined the effectiveness of pairs trading strategies in cryptocurrency markets and found that while traditional methods underperform at daily frequencies, high-frequency trading (at 5-minute intervals) significantly improves profitability, highlighting the market's inefficiencies and the importance of intervals at which data are collected. This is possible in the case of Cryptos since Cryptos trade 24 hours a day, seven days a week. Tadi and Kortchemski (2021) proposed a dynamic cointegration-based pairs trading strategy in cryptocurrency markets, demonstrating that incorporating optimal look-back windows and realistic order execution modeling significantly enhances risk-adjusted returns compared to naive buy-and-hold approaches. Saji (2021) investigated the profitability of pairs trading in cryptocurrency markets using four statistical methods, correlation analysis, distance approach, stochastic return differential and cointegration, and found that the long-short strategies consistently outperform long-only portfolios, challenging the notion of market efficiency in crypto trading . Ardia et al. (2016) demonstrated that incorporating orthogonal regression through Bayesian normalization significantly enhances the accuracy of spread estimation in pairs trading, leading to improved risk-adjusted returns and more robust strategy

performance . Kim and Kim (2019) incorporated orthogonal regression into their deep reinforcement learning-based pairs trading framework, demonstrating that it yields more consistent hedge ratios and superior trading signals than ordinary least squares, thereby enhancing the profitability and robustness of the strategy.

Our analysis here begins by identifying cointegrated pairs using both standard and reverse-order regression frameworks. We instead use the daily data. The primary focus is on three key pairs: Bitcoin and Ethereum, Ethereum and Ethereum Classic, Bitcoin and Ethereum Classic. For each pair, the cointegration relationship is tested using the Augmented Dickey-Fuller (ADF) test, and the residual spread series is analyzed for mean-reverting behavior.

Once cointegrated pairs are established, trading strategies are constructed based on the threshold rules applied to the spread series. These strategies are evaluated under various configurations, including different entry and exit thresholds and alternative spread definitions. To enhance signal precision, Orthogonal Distance Regression (ODR) is employed as an alternative to Ordinary Least Squares (OLS) in estimating the cointegration relationship, accounting for measurement errors in the presence of both cryptocurrencies in the pair and providing a more balanced estimation of the underlying relationship.

The profitability of the proposed strategies is evaluated using historical data and out-of-sample testing, allowing us to assess their performance under realistic market conditions. This comprehensive framework provides empirical evidence on the effectiveness of pair trading in the cryptocurrency market and offers practical insights for quantitative trading and risk management. In addition to standard cointegration analysis, we explore the impact of reversing the regression order and adjusting the spread thresholds. These variations help determine the sensitivity of the strategy to model

specifications and parameter choices. Finally, we implement the pair-trading framework using the separate training and test datasets to evaluate its generalizability and predictive power.

6.2 Cointegration Analysis

This section presents the cointegration analysis for selected cryptocurrency pairs. This kind of analysis serves as the foundation for constructing spread-based pair-trading strategies, with that it evaluates the suitability of the pair under consideration. Identifying a cointegration relationship enables the creation of a mean-reverting spread series, which is essential for developing the rules for the effective pair-trading. We focus on three prominent cryptocurrency pairs: Bitcoin and Ethereum; Ethereum and Ethereum Classic; and Bitcoin and Ethereum Classic. For each pair, we perform the linear regression using the logarithmic prices of the two assets under consideration to estimate the cointegration relationship.

Let P_{it} denote the observed price of cryptocurrency i at time t, and define the log price as $p_{it} = \ln(P_{it})$. Under the assumption that the two cryptocurrencies exhibit similar risk profiles, their log prices p_{1t} and p_{2t} are expected to follow a common trend and be cointegrated based on the APT. To examine this, we consider the following linear regression model

$$p_{1t} = \beta_0 + \beta_1 p_{2t} + \varepsilon_t \tag{6.1}$$

where β_0 and β_1 are unknown regression parameters, and ε_t denotes the random error term. We assume that the errors are randomly and independently distributed as normal with $E(\varepsilon_t) = 0$ and constant variance $Var(\varepsilon_t) = \sigma^2$. The residual series r_t is computed as

$$r_t = p_{1t} - \hat{\beta}_0 - \hat{\beta}_1 p_{2t} \tag{6.2}$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are the least-squares estimates of the regression coefficients. And define the corresponding spread series as

$$w_t = p_{1t} - \hat{\beta}_1 p_{2t} = \hat{\beta}_0 + r_t \tag{6.3}$$

To test for cointegration, we first conduct the preliminary diagnostic by examining the time series plot and the ACF plot of the residuals, as discussed in Chapter 1. Subsequently, we apply the Augmented Dickey-Fuller (ADF) test to formally assess stationarity.

Let r_t represent the residual series of interest, modeled as an autoregressive model of order p, denoted AR(p). The ADF test involves estimating the following regression

$$r_t = c_t + \beta r_{t-1} + \sum_{i=1}^{p} \phi_i \Delta r_{t-i} + e_t \tag{6.4}$$

where $\Delta x_j = x_j - x_{j-1}$ is the first-differenced series, and c_t is a deterministic component that may be zero, a constant, or a linear trend. The null hypothesis $H_0 : \beta = 1$ (presence of unit root) is tested against the alternative $H_a : \beta < 1$. The test statistic is given by

$$\text{ADF-test} = \frac{\hat{\beta} - 1}{\hat{\text{std}}(\hat{\beta})} \tag{6.5}$$

where $\hat{\beta}$ is the least-squares estimate of β. A rejection of the null hypothesis indicates that the residual series is stationary, thereby confirming the existence of a cointegration relationship between the asset pair.

6.2.1 Ethereum - Bitcoin

We begin the cointegration analysis by examining the relationship between Ethereum and Bitcoin, two of the most widely traded and most influential cryptocurrencies. As indicated earlier, the analysis is conducted using the logarithmic

price series of both assets to assess whether a long-run equilibrium relationship exists. Here p_{1t} is the log price of Ethereum at time t and p_{2t} is the log price of Bitcoin at time t. We will regress p_{1t} on p_{2t}.

The Pearson correlation coefficient between the raw daily closing prices of Bitcoin and Ethereum is 0.930219, indicating a strong linear association. After applying a logarithmic transformation to the price series, the correlation increases only slightly to 0.9381054. This suggests that the two assets exhibit highly synchronized price movements, reinforcing the hypothesis that they may share a stable long-term relationship suitable for pair-trading.

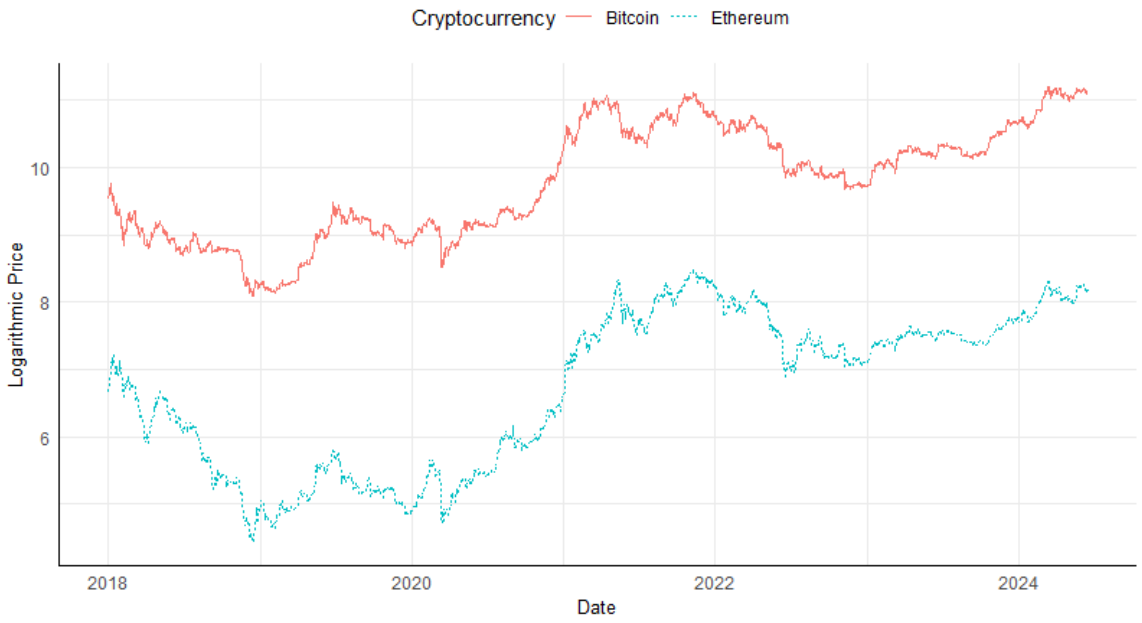


Figure 6.1: Logarithmic Prices of Ethereum and Bitcoin Over Time

Given this strong correlation, we proceed to test for cointegration between the log price series. If cointegration is confirmed, it implies that the spread between the two assets is mean-reverting, which is a key requirement for constructing a profitable pair-trading strategy. The time plot of the daily log prices of Ethereum and Bitcoin is shown in Figure 6.1, suggesting their co-movement over time.

We thus fit a linear regression model with the log price of Ethereum (p_{1t}) as the dependent variable and the log price of Bitcoin p_{2t} as the independent variable, of the form

$$p_{1t} = \beta_0 + \beta_1 p_{2t} + \varepsilon_t,$$

where ε_t represents the random error term. The corresponding spread series w_t $(= p_{1t} - \hat{\beta}_1 p_{2t})$ are interpreted as the spread-variable between Ethereum and Bitcoin. Since the roles of dependent and independent variables have been specified we call it Ethereum–Bitcoin spread. The corresponding spread when the roles are interchanged will be called Bitcoin–Ethereum spread.

Table 6.1: Linear Regression of Ethereum on Bitcoin

Variable	**Estimate**	**Std. Error**	**t value**	**P value**
$\hat{\beta}_0$	-5.916162	0.096309	-61.43	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (Bitcoin)	1.289980	0.009808	131.53	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.3995 on 2358 degrees of freedom				
Multiple R-squared: 0.88				
Adjusted R-squared: 0.88				
F-statistic: 1.73×10^4 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

From the regression analysis summarized in Table 6.1, the coefficient for $\hat{\beta}_1$ (Bitcoin) is highly significant, with a p-value less than 2×10^{-16}, suggesting a strong linear relationship between the log prices of Bitcoin and Ethereum. The model explains approximately 88% of the variation in Ethereum's log price, as indicated by the high R-squared value. The estimated cointegration model between the log prices of Ethereum and Bitcoin is given by

$$\hat{p}_{1t} = -5.916162 + 1.289980\, p_{2t}$$

The residual standard deviation is estimated to be $\hat{\sigma}_\varepsilon = 0.3995$, indicating moderate dispersion around the fitted equilibrium.

To evaluate the mean-reverting behavior of the residuals r_t, we examine their time series and autocorrelation structure. Figure 6.2 displays the residuals over time, revealing fluctuations around a stable mean albeit with occasional large deviations. The Figure 6.3 presents the sample Autocorrelation function of the residual series. The presence of significant autocorrelations at lags of shot terms as well as medium time frame support the hypothesis that the residuals exhibit temporal dependence.

To formally assess the presence of cointegration, we apply the Augmented Dickey-Fuller (ADF) test to the residual series r_t obtained from the regression model. In this analysis, we apply the ADF test with no constant or trend component (type = none), as cointegration residuals are expected to fluctuate around a zero mean without drift. Thus, the ADF test model 6.4 is simplified to

$$r_t = \beta r_{t-1} + \sum_{i=1}^{p} \phi_i \Delta r_{t-i} + e_t \tag{6.6}$$

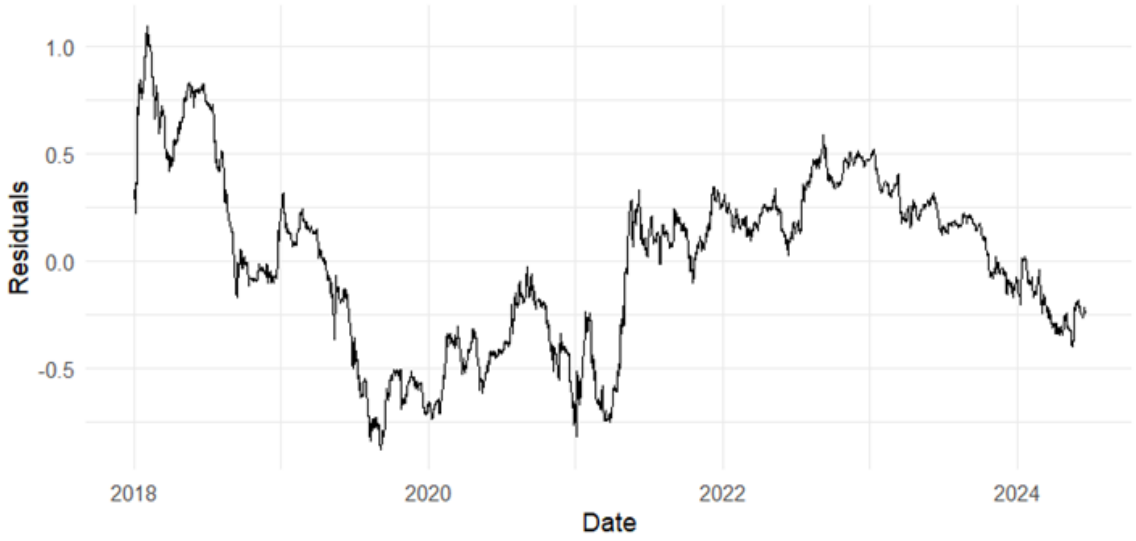


Figure 6.2: Residual Series from Regression of Ethereum on Bitcoin

The optimal number of lags is selected using the Akaike Information Criterion (AIC), which balances model fit and complexity. Based on AIC, the test selects lag of one. Therefore, the ADF test model being estimated is

$$r_t = \beta r_{t-1} + \phi_1 \Delta r_{t-1} + e_t \tag{6.7}$$

Rejection of the null supports the conclusion that the residuals are stationary, thereby confirming cointegration between the log price series of Bitcoin and Ethereum.

The results of the ADF test are summarized in Table 6.2. The test statistic is -1.7995, with corresponding P-value = 0.0721. Thus, at the 5% level, we fail to reject the null hypothesis of a unit root at that threshold. However, the p-value of 0.0721 suggests marginal significance at the 10% level, indicating weak evidence of stationarity. This result implies that the spread series exhibits marginally significant mean-reverting

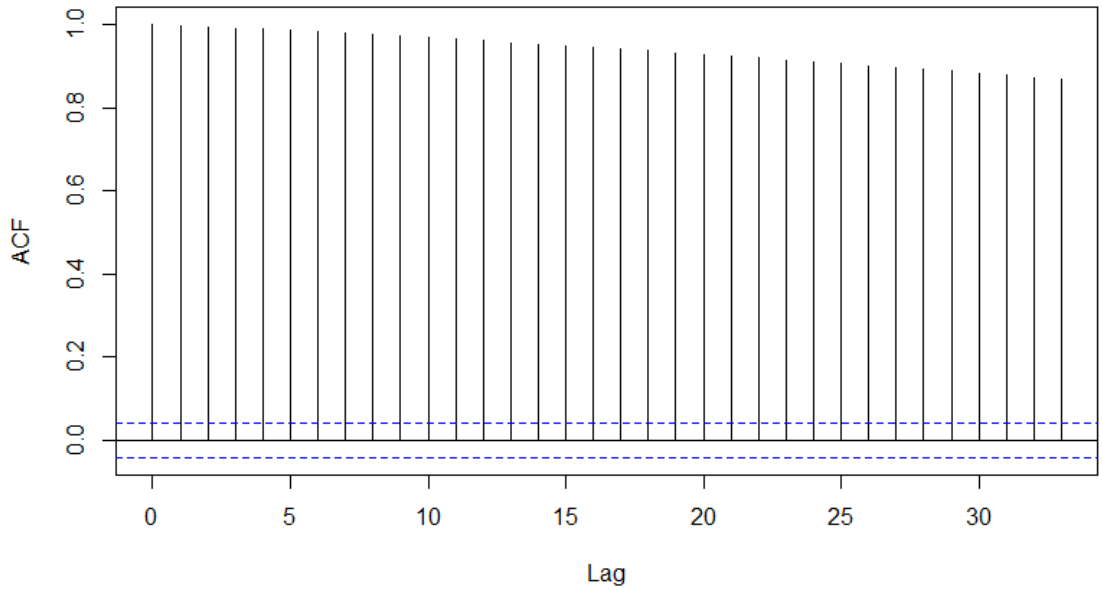


Figure 6.3: ACF Analysis of Residuals from Regression of Ethereum on Bitcoin

Table 6.2: Augmented Dickey-Fuller Test for Residuals from Regression of Ethereum on Bitcoin

Coefficient	**Estimate**	**Std. Error**	**t value**	**P value**
$\beta - 1$	-0.002523	0.001402	-1.799	0.072072
ϕ_1	0.068956	0.020555	3.355	0.000807
Model Statistics				
Residual standard error: 0.02718 on 2356 degrees of freedom				
Multiple R-squared: 0.005951				
Adjusted R-squared: 0.005107				
F-statistic: 7.052 on 2 and 2356 DF				
p-value: 0.0008843				
Value of test-statistic is: -1.7995				
Critical values for test statistics: 1%: -2.58, 5%: -1.95, 10%: -1.62				

behavior, and the cointegration relationship is supported under relaxed significance criteria.

6.2.2 Ethereum - Ethereum Classic

For the pair of Ethereum and Ethereum Classic, the Pearson correlation coefficient of their daily closing prices is 0.7938, and the correlation for their daily log prices is even higher at 0.881. These high correlations suggest that the two cryptocurrencies likely share similar underlying market dynamics and may exhibit comparable long-term price trends.

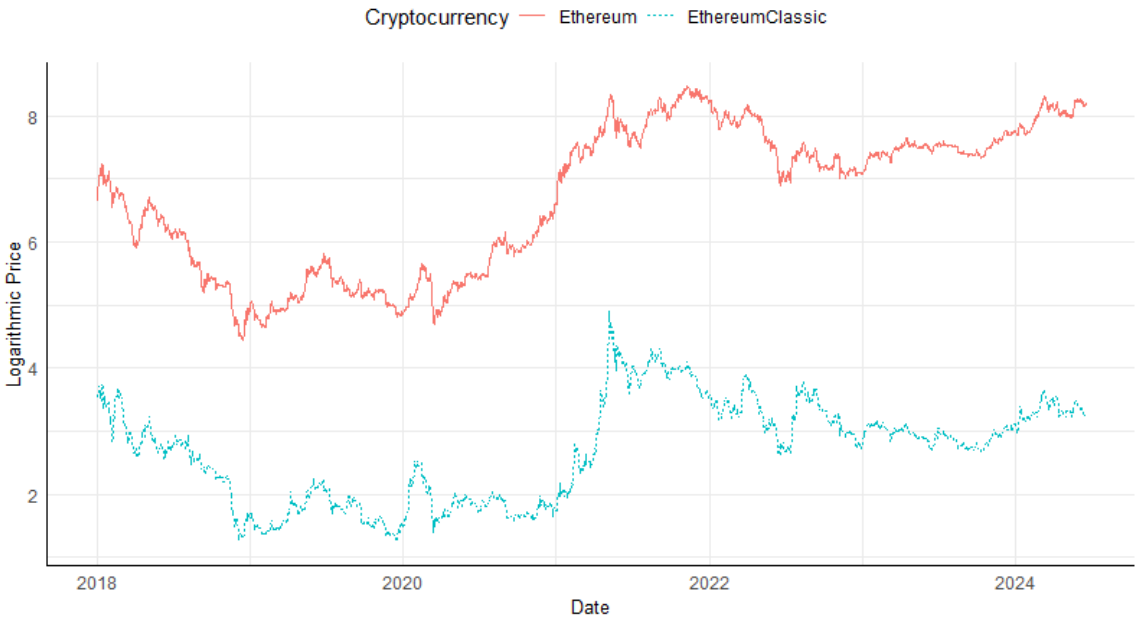


Figure 6.4: Logarithmic Prices of Ethereum and Ethereum Classic Over Time

The Figure 6.4 illustrates the logarithmic price curves of Ethereum and Ethereum Classic over the study period. The plot reveals a consistent co-movement between the two

cryptocurrencies, with Ethereum generally maintaining a higher price level. Despite short-term fluctuations, the overall patterns of rise and fall in both assets appear synchronized, reinforcing the hypothesis of a shared stochastic trend.

Table 6.3: Linear Regression of Ethereum on Ethereum Classic

Variable	**Estimate**	**Std. Error**	**t value**	**P value**
$\hat{\beta}_0$	3.27135	0.03958	82.65	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (EthereumCls)	1.27403	0.01408	90.46	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.5456 on 2358 degrees of freedom				
Multiple R-squared: 0.7763				
Adjusted R-squared: 0.7762				
F-statistic: 8183 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

To formally assess the long-term relationship, we perform a linear regression using the logarithmic price of Ethereum as the dependent variable and that of Ethereum Classic as the independent variable. The regression demonstrates a well fit, with an R^2 value of 0.7763 and a highly significant F-statistic (p-value $< 2.2 \times 10^{-16}$), as shown in Table 6.3. The estimated model is

$$\hat{p}_{1t} = 3.27135 + 1.27403\, p_{2t}, \quad \hat{\sigma}_\varepsilon = 0.5456$$

where p_{1t} and p_{2t} denote the log prices of Ethereum and Ethereum Classic respectively at time t, and $\hat{p}_{1t}$ represents the estimate of p_{1t}.

Figure 6.5: Residual Series from Regression of Ethereum on Ethereum Classic

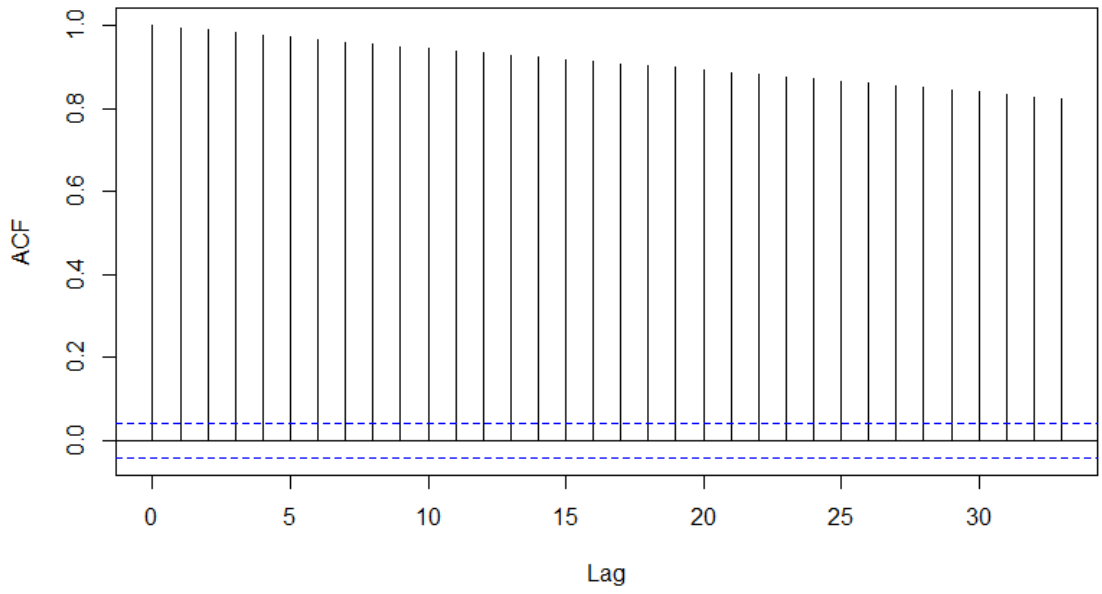


Figure 6.6: ACF Analysis of Residuals from Regression of Ethereum on Ethereum Classic

The residuals r_t are then computed and analyzed to assess stationarity. The Figure 6.5 displays the time series plot of the residuals, revealing fluctuations around a stable mean, while Figure 6.6 presents the sample ACF, which shows a gradual decay. These patterns suggest potential mean-reverting behavior in the spread series, supporting the hypothesis of cointegration between the two assets.

Table 6.4: Augmented Dickey-Fuller Test for Residuals from Regression of Ethereum on Ethereum Classic

Coefficient	**Estimate**	**Std. Error**	**t value**	**P value**
$\beta - 1$	-0.004453	0.001804	-2.468	0.0137
ϕ_1	0.100543	0.020472	4.911	9.67×10^{-7}
Model Statistics				
Residual standard error: 0.04769 on 2356 degrees of freedom				
Multiple R-squared: 0.01229				
Adjusted R-squared: 0.01145				
F-statistic: 14.65 on 2 and 2356 DF				
p-value: 4.729×10^{-7}				
Value of test-statistic: -2.468				
Critical values: 1%: -2.58, 5%: -1.95, 10%: -1.62				

As earlier, we apply the Augmented Dickey-Fuller test to assess whether the series is stationary. Based on the AIC, the optimal lag length selected for the test is one; therefore, the ADF model remains the same and is given by the specification in Equation 6.7 The computed ADF test statistic is -2.468, and the corresponding p-value is 0.0137. Therefore, we reject the null hypothesis of a unit root at the 5% significance level. The detailed results are summarized in Table 6.4. This outcome provides statistical evidence

that the residual series is stationary, supporting the presence of a cointegration relationship between Ethereum and Ethereum Classic.

6.2.3 Bitcoin - Ethereum Classic

We further explore the potential cointegration relationship between Bitcoin and Ethereum Classic. The Pearson correlation coefficient for their daily closing prices is 0.6449, and for their daily log prices, it increases to 0.7461, indicating a moderate to strong linear association. These values suggest that the two cryptocurrencies may exhibit partially synchronized price movements over time.

The time series plot of their daily log prices is shown in Figure 6.7. While Bitcoin consistently maintains a higher price level, the overall trends of both assets show a degree of co-movement, with similar directional shifts during major market events. This visual evidence supports the hypothesis of a shared underlying trend, suggesting further investigation through cointegration analysis.

Table 6.5: Linear Regression of Bitcoin on Ethereum Classic

Variable	**Estimate**	**Std. Error**	**t value**	**P value**
$\hat{\beta}_0$	7.66954	0.04052	189.26	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (EthereumCls)	0.78453	0.01442	54.41	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.5586 on 2358 degrees of freedom				
Multiple R-squared: 0.5566				
Adjusted R-squared: 0.5564				
F-statistic: 2960 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

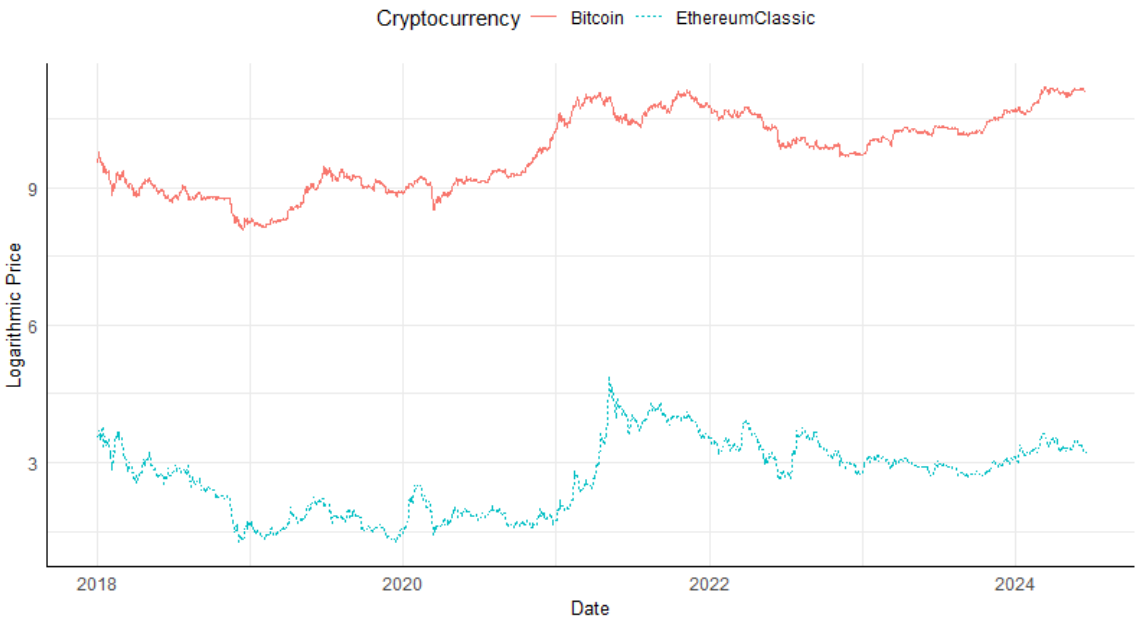


Figure 6.7: Logarithmic Prices of Bitcoin and Ethereum Classic Over Time

To formally evaluate the relationship between Bitcoin and Ethereum Classic, we perform a linear regression using Bitcoin's log prices as the dependent variable and Ethereum Classic's log prices as the independent variable. The regression results are presented in Table 6.5. The estimated model is given by

$$\hat{p}_{1t} = 7.66954 + 0.78453\, p_{2t}, \quad \hat{\sigma}_\varepsilon = 0.5586$$

where p_{1t} and p_{2t} denote the log prices of Bitcoin and Ethereum Classic respectivelyat time t, and $\hat{p}_{1t}$ represents the estimate of p_{1t}. The model explains approximately 55.66% of the variation in Bitcoin's log price, with a highly significant F-statistic and a p-value less than 2.2×10^{-16}, indicating a statistically meaningful relationship.

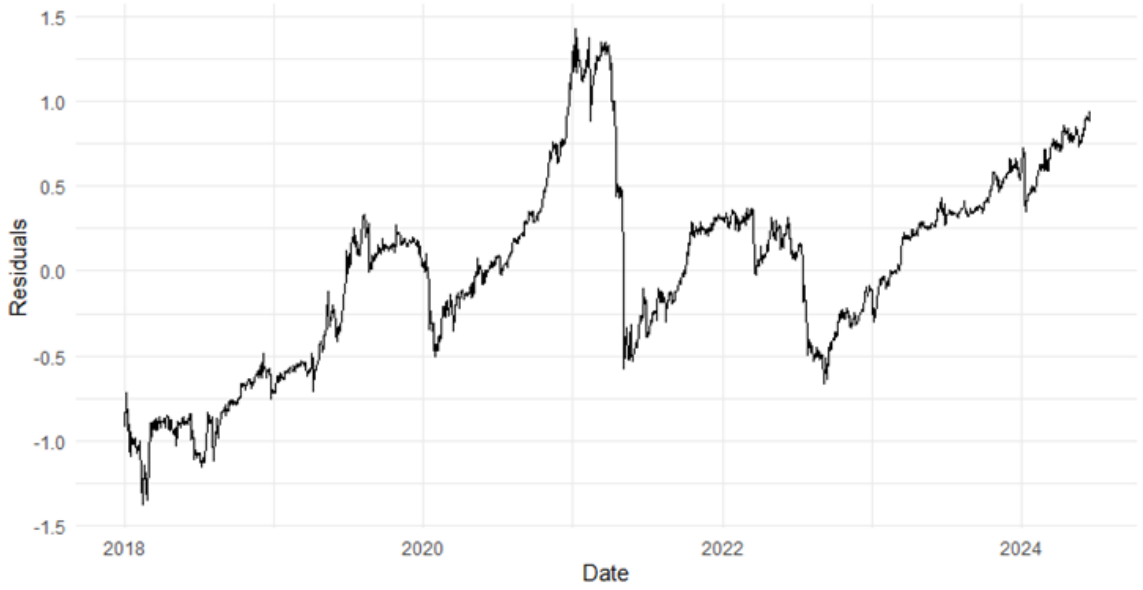


Figure 6.8: Residual Series from Regression of Bitcoin on Ethereum Classic

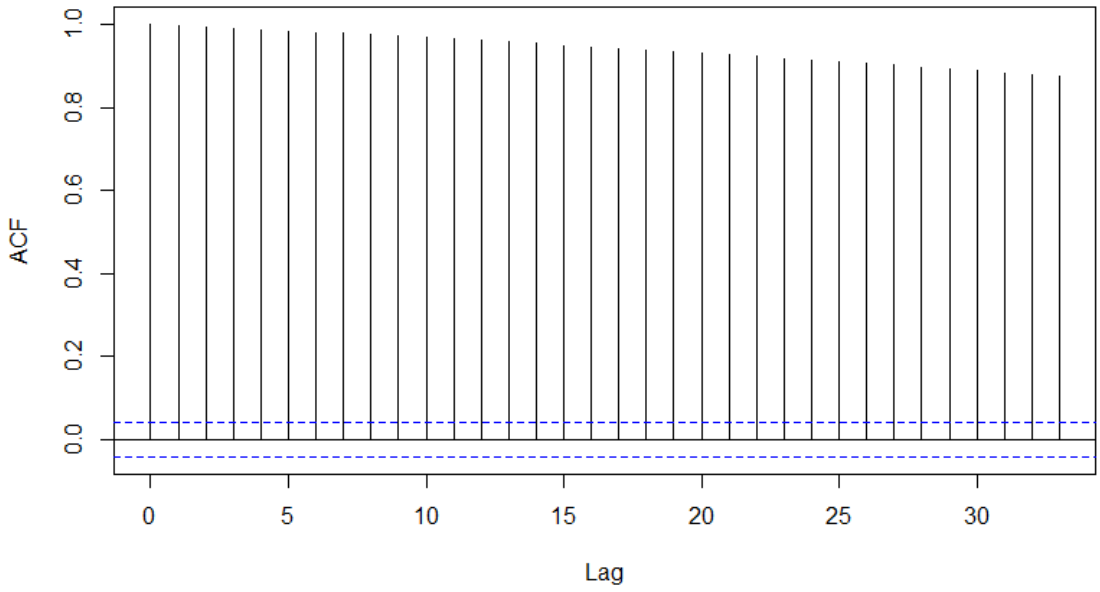


Figure 6.9: ACF Analysis of Residuals from Regression of Bitcoin on Ethereum Classic

Based on the estimated model, the residual series r_t is computed and plotted over time in the Figure 6.8, and the autocorrelation structure is examined via the sample ACF in Figure 6.9. The residual series exhibits fluctuations around a stable mean, and the ACF shows a slow but gradual decay.

Table 6.6: Augmented Dickey-Fuller Test for Residuals from Regression of Bitcoin on Ethereum Classic

Coefficient	**Estimate**	**Std. Error**	**t value**	**P value**
$\beta - 1$	-0.001848	0.001216	-1.520	0.129
ϕ_1	0.114966	0.020450	5.622	2.11×10^{-8}
Model Statistics				
Residual standard error: 0.03292 on 2356 degrees of freedom				
Multiple R-squared: 0.014				
Adjusted R-squared: 0.01316				
F-statistic: 16.72 on 2 and 2356 DF				
p-value: 6.158×10^{-8}				
Value of test-statistic: -1.5196				
Critical values: 1%: -2.58, 5%: -1.95, 10%: -1.62				

The Augmented Dickey-Fuller Test is then performed on the residual series to assess stationarity. Using the Akaike Information Criterion, the optimal lag length is determined to be one; therefore, the ADF model retains the same structure as in Equation 6.7. As shown in Table 6.6, the test statistic is -1.5196 with a corresponding p-value of 0.129, we fail to reject the null hypothesis of a unit root at conventional significance levels, indicating that the spread between Bitcoin and Ethereum Classic does not exhibit strong mean-reverting behavior under strict statistical criteria. However, the test statistic is close to the 10% critical value, and the residual series shows visual signs of mean reversion in

its time plot and autocorrelation structure. Despite the lack of formal cointegration, we will still consider this pair for pair trading to evaluate its practical performance.

6.3 Cointegration Analysis When the Roles are Reversed

We now conduct the above analyses by interchanging the roles of the dependent and independent variables of previous subsection. This approach provides an alternative spread series in each case, which can be used to construct pair trading strategies. Further, by analyzing both directions of the relationship, we may be able to gain the flexibility to initiate trades in either direction, buying one asset while shorting the other, depending on whether the spread deviates above or below its long-run equilibrium. This bidirectional framework will be utilized in the next section to implement and evaluate trading strategies. To distinguish the setup from the previous one, we will add an asterisk in the notation, and accordingly the notation used are p^*_{1t}, p^*_{2t}, w^*_t, $\hat{\sigma}^*_\varepsilon$, and so forth.

6.3.1 Bitcoin - Ethereum

We begin by conducting a regression analysis between Bitcoin and Ethereum, in which the logarithmic price of Bitcoin is modeled as the dependent variable and that of Ethereum as the independent variable. Accordingly we call the corresponding spread as Bitcoin - Ethereum spread. The regression results are presented in Table 6.7.

Based on the regression results, the estimated model is

$$\hat{p}^*_{1t} = 5.209722 + 0.682213\, p^*_{2t}, \quad \hat{\sigma}^*_\varepsilon = 0.2905,$$

where p^*_{1t} and p^*_{2t} denote the logarithmic prices of Bitcoin and Ethereum respectively at time t, and $\hat{p}^*_{1t}$ represents the estimate of p^*_{1t}. The model explains approximately 88% of

Table 6.7: Linear Regression of Bitcoin on Ethereum

Variable	Estimate	Std. Error	t value	P value
$\hat{\beta}_0$	5.209722	0.035287	147.6	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (Ethereum)	0.682213	0.005187	131.5	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.2905 on 2358 degrees of freedom				
Multiple R-squared: 0.88				
Adjusted R-squared: 0.88				
F-statistic: 1.73×10^4 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

the variation in Bitcoin's log price, with a highly significant F-statistic. The resulting spread series is defined as

$$w_t^* = p_{1t}^* - 0.682213\, p_{2t}^*$$

The spread w_t^* captures the deviation from the estimated equilibrium relationship and serves as the basis for generating trading signals. For trading purposes, we select a threshold based on the residual standard error value for pair trading strategies.

The stationarity of the residual series is evaluated using the ADF test, with the optimal lag length of one selected by the AIC. Consequently, the test model remains the same as in the previous section, and the results are summarized in Table 6.8. The test statistic is -1.7199 with a corresponding p-value of 0.08557. Therefore, we reject the null hypothesis of a unit root at the 10% significance level, indicating weak evidence of stationarity. The result suggests that deviations from the long-run equilibrium may exhibit mean-reverting behavior, supporting the use of the spread in pair trading strategies under relaxed statistical assumptions.

Table 6.8: Augmented Dickey-Fuller Test for Residuals from Regression of Bitcoin on Ethereum

Coefficient	Estimate	Std. Error	t value	P value
$\beta - 1$	-0.002475	0.001439	-1.720	0.08557
ϕ_1	0.065063	0.020565	3.164	0.00158
Model Statistics				
Residual standard error: 0.02028 on 2356 degrees of freedom				
Multiple R-squared: 0.005314				
Adjusted R-squared: 0.00447				
F-statistic: 6.293 on 2 and 2356 DF				
p-value: 0.00188				
Test Statistics				
Value of test-statistic: -1.7199				
Critical values for test statistics: 1%: -2.58, 5%: -1.95, 10%: -1.62				

6.3.2 Ethereum Classic - Ethereum

In the regression model of reversed roles for Ethereum and Ethereum Classic, the logarithmic price of Ethereum is used as the independent variable, while the logarithmic price of Ethereum Classic serves as the dependent variable. The results of the linear regression analysis are presented in Table 6.9. The model exhibits a high degree of explanatory power, with a R^2 of 0.7763 and a highly significant F-statistic.

Based on the regression results, the estimated spread series is defined as

$$w_t^* = p_{1t}^* - 0.609331\, p_{2t}^*$$

where p_{1t}^* and p_{2t}^* denote the logarithmic prices of Ethereum Classic and Ethereum respectively at time t.

To evaluate the stationarity of this spread, we apply the ADF test with the optimal lag length of one selected by the Akaike Information Criterion and hence the same test

Table 6.9: Linear Regression of Ethereum Classic on Ethereum

Variable	**Estimate**	**Std. Error**	**t value**	**P value**
$\hat{\beta}_0$	-1.390518	0.045824	-30.34	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (Ethereum)	0.609331	0.006736	90.46	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.3773 on 2358 degrees of freedom				
Multiple R-squared: 0.7763				
Adjusted R-squared: 0.7762				
F-statistic: 8183 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

Table 6.10: Augmented Dickey-Fuller Test for Residuals from Regression of Ethereum Classic on Ethereum

Coefficient	**Estimate**	**Std. Error**	**t value**	**P value**
$\beta - 1$	-0.006705	0.002176	-3.081	0.00209
ϕ_1	0.074378	0.020522	3.624	0.00030
Model Statistics				
Residual standard error: 0.03978 on 2356 degrees of freedom				
Multiple R-squared: 0.0091				
Adjusted R-squared: 0.0083				
F-statistic: 10.84 on 2 and 2356 DF				
p-value: 2.06×10^{-5}				
Test Statistics				
Value of test-statistic: -3.081				
Critical values for test statistics: 1%: -2.58, 5%: -1.95, 10%: -1.62				

model as in the previous section. The results, shown in Table 6.10, indicate that the test statistic is -3.081 with a corresponding statistically significant p-value of 0.00209. This leads to the rejection of the null hypothesis of a unit root at the 1% significance level. These findings provide strong evidence that the spread series is stationary, supporting the

existence of a long-run equilibrium relationship between the log prices of Ethereum Classic and Ethereum.

6.3.3 Ethereum Classic - Bitcoin

For the regression model in which the logarithm of Ethereum Classic's price is expressed as a function of the logarithm of Bitcoin's price, the result is summarized in the Table 6.11.

Table 6.11: Linear Regression of Ethereum Classic on Bitcoin

Variable	**Estimate**	**Std. Error**	**t value**	**P value**
$\hat{\beta}_0$	-4.24662	0.12805	-33.16	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (Bitcoin)	0.70949	0.01304	54.41	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.5312 on 2358 degrees of freedom				
Multiple R-squared: 0.5566				
Adjusted R-squared: 0.5564				
F-statistic: 2960 on 1 and 2358 DF				
p-value: $< 2.2 \times 10^{-16}$				

Based on the regression output, the estimated spread series is given by

$$w_t^* = p_{1t}^* - 0.70949\, p_{2t}^*$$

where p_{1t}^* and p_{2t}^* represent the logarithmic prices of Ethereum Classic and Bitcoin at time t, respectively. The term w_t^* denotes the spread series, capturing the deviation from the estimated long-run equilibrium relationship between the two assets.

Table 6.12: Augmented Dickey-Fuller Test for Residuals from Regression of Ethereum Classic on Bitcoin

Coefficient	**Estimate**	**Std. Error**	**t value**	**P value**
$\beta - 1$	-0.004136	0.001690	-2.447	0.01448
ϕ_1	0.067468	0.020545	3.284	0.00104
Model Statistics				
Residual standard error: 0.04353 on 2356 degrees of freedom				
Multiple R-squared: 0.006854				
Adjusted R-squared: 0.006011				
F-statistic: 8.129 on 2 and 2356 DF				
p-value: 0.0003031				
Test Statistics				
Value of test-statistic: -2.4469				
Critical values for test statistics: 1%: -2.58, 5%: -1.95, 10%: -1.62				

To assess the stationarity of the spread, we conduct an ADF test with the optimal lag length of one selected by the Akaike Information Criterion (AIC). Consequently, the test model remains the same as in the previous section. As shown in Table 6.12, the test statistic is -2.447 with the corresponding significant p-value of 0.01448, indicating that the null hypothesis of a unit root can be rejected at the 5% significance level. This suggests that the residual series is stationary, supporting the presence of a cointegrating relationship between Ethereum Classic and Bitcoin.

We summarize the results of the Augmented Dickey–Fuller tests for various cryptocurrency pairs in Table 6.13. The direction of regression in cointegration testing can influence the stationarity of the residual spread series. For instance, in the Bitcoin–Ethereum Classic pair, using the log of Bitcoin price as the dependent variable and the log of Ethereum Classic price as the independent variable yields an ADF test statistic of -1.5196 with a p-value of 0.129, indicating non-stationarity. However, reversing the regression direction results in an ADF test statistic of -2.4469 with a

Table 6.13: Augmented Dickey-Fuller Test Results for Spread Pairs

Spread Pair	**Test Statistic**	**P-value**	**Cointegration Result**
Ethereum - Bitcoin	-1.799	0.0721	Marginally cointegrated
Bitcoin - Ethereum	-1.720	0.0856	Marginally cointegrated
Ethereum - Ethereum Classic	-2.468	0.0137	Cointegrated
Ethereum Classic - Ethereum	-3.081	0.0021	Cointegrated
Bitcoin - Ethereum Classic	-1.520	0.1290	Not cointegrated
Ethereum Classic - Bitcoin	-2.447	0.0145	Cointegrated

p-value of 0.01448, confirming stationarity. This asymmetry arises because cointegration is not inherently symmetric, and ordinary least squares (OLS) regression is directional. Reversing the regression may not preserve this property, as OLS minimizes the sum of squared residuals for the dependent variable. The residuals from the reverse regression can behave very differently as they correspond to horizontal signed distances in the original model.

6.4 Pair Trading strategy

Building upon the cointegration analysis conducted in the previous section, we develop spread-based trading strategies for cryptocurrency pairs that exhibit statistically significant long-run equilibrium relationships. These strategies are founded on the assumption that cointegrated assets tend to revert to their equilibrium over time, enabling opportunities for market-neutral arbitrage. For each identified pair, the spread series is derived from the residuals of the regression. Trading signals are generated when the spread deviates significantly from its equilibrium level, with thresholds defined using the estimated standard error of the residuals. This approach allows for systematic entry and exit rules based on statistical deviations, facilitating the implementation of robust and repeatable trading strategies.

6.4.1 Pair Trading with Ethereum and Bitcoin

We begin by constructing a pair trading strategy for Ethereum and Bitcoin, using the cointegration relationship identified in the preceding analysis. In the first formulation, the logarithmic price of Ethereum is modeled as the dependent variable, and that of Bitcoin as the independent variable (denoted by notation Ethereum - Bitcoin). From the regression results, the spread series is given by

$$w_t = p_{1t} - 1.289980\, p_{2t}$$

where p_{1t} and p_{2t} denote the logarithmic prices of Ethereum and Bitcoin at time t, respectively. The standard error of the spread series w_t is estimated to be 0.3995. Accordingly, we select the threshold parameter $\Delta = 0.4$, which slightly exceeds the residual standard error to allow for meaningful deviations from equilibrium.

We had concluded the spread series Ethereum - Bitcoin is stationary, with an estimated mean of $\mu_w = \hat{\beta}_0 = -5.91616$. To visualize this, Figure 6.10 presents the time series plot of w_t, constructed from the daily log prices of Ethereum and Bitcoin. The plot includes horizontal reference lines corresponding to the mean μ_w, as well as the upper and lower thresholds $\mu_w + \Delta$ and $\mu_w - \Delta$, respectively. The figure illustrates the dynamic behavior of the spread series. Periods where the above spread crosses below the lower threshold suggest potential entry points for long Ethereum and short Bitcoin positions, anticipating a reversion toward the mean. Conversely, when the spread approaches or exceeds the upper threshold, the strategy signals an exit.

To implement the trading strategy, we extend the dataset to include an out-of-sample period spanning from June 18, 2024, to February 27, 2025. Historical daily

Figure 6.10: Spread Series Over Time for Ethereum - Bitcoin

price data for Bitcoin and Ethereum are retrieved from the websites Ethereum (ETH) Historical Data | CoinCodex [1], Bitcoin (BTC) Historical Data | CoinCodex [2].

The newly acquired data are merged with the original dataset used to construct the trading strategy. The spread series for the extended dataset is computed using the same regression-based formula. The Figure 6.11 displays the time series plot of the spread over the full sample period, including both in-sample and out-of-sample data, illustrating the dynamics over the combined dataset period. During the out-of-sample period, the spread crosses below the lower threshold, indicating a potential entry point for a long Ethereum and short Bitcoin position.

[1] https://coincodex.com/crypto/ethereum/historical-data/
[2] https://coincodex.com/crypto/bitcoin/historical-data/

Figure 6.11: Spread Series for Ethereum - Bitcoin with Out-of-sample Period

The trading strategy is defined as follows

- Entry Condition: At time t, initiate a long position in one dollar of Ethereum and a short position in $\hat{\beta}_1 = 1.289980$ dollars of Bitcoin when the spread $w_t = \mu_w - \Delta$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) when the spread $w_{t+1} = \mu_w + \Delta$.

The first trade is initiated in June 2019 when the spread falls below the lower threshold, and is closed in August 2022 upon reversion to the upper threshold. A second entry signal is observed in August 2024, indicating another opportunity for arbitrage. To evaluate the actual return from the trading strategy, we simulate a position initiated with $1000 invested in Ethereum and a simultaneous short position of $1289.98 in Bitcoin. The

Table 6.14 summarizes the trade details, including entry and exit dates, asset prices, the number of units held, and the resulting return. The first trade yields a total return of $3187.20. These results demonstrate the practical viability of the pair trading strategy when applied to real market data.

Table 6.14: Trading Strategy Details - Ethereum Long and Bitcoin Short

Date	**Action**	**Price Ethereum**	**Price Bitcoin**	**Long Units Ethereum**	**Short Units Bitcoin**	**Return**
2019-06-25	Entry	318.12647	11,790.92	3.1434	0.1094	
2022-08-05	Exit	1,732.2545	23,289.32			3,187.1989
2024-08-07*	Entry	2,341.4742	55,024.12	0.4271	0.0234	

* Occurs during the Out-of-sample period.

For the reversed-order model, the spread series is defined as

$$w_t^* = p_{1t}^* - 0.682213\, p_{2t}^*$$

where p_{1t}^* and p_{2t}^* denote the logarithmic prices of Bitcoin and Ethereum at time t, respectively.

The Figure 6.12 presents the time series plot of w_t^*, constructed from the daily log prices of Bitcoin and Ethereum. To facilitate interpretation, the plot includes horizontal reference lines corresponding to the mean of the spread, $\mu_w^* = \hat{\beta}_0^* = 5.209722$, as well as the upper and lower thresholds $\mu_w^* + \Delta^*$ and $\mu_w^* - \Delta^*$, respectively. The Δ^* is selected to be 0.3 that is slightly larger than the standard error $\sigma_w^* = \hat{\sigma}_\varepsilon^* = 0.2905$. The spread continues to exhibit mean-reverting characteristics, with fluctuations around the long-run

equilibrium. The threshold lines provide a visual framework for identifying potential trading signals, where deviations beyond the bounds may indicate entry or exit opportunities.

Figure 6.12: Spread Series Over Time for Bitcoin - Ethereum

To implement the trading strategy, we extend, as earlier, the dataset to include the out-of-sample period from June 18, 2024, to February 27, 2025. The spread for the new period is calculated using the same formula as before. A time series plot of the resulting spread is presented in Figure 6.13, illustrating its behavior over the full sample period.

The trading strategy is defined as follows

Figure 6.13: Spread Series for Bitcoin - Ethereum with Out-of-sample Period

- Entry Condition: At time t, initiate a long position in Bitcoin for one dollar and a short position in 0.682213 dollars of Ethereum when the spread $w_t^* = \mu_w^* - \Delta^*$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) when the spread $w_{t+i}^* = \mu_w^* + \Delta^*$.

This strategy is implemented on the dataset to identify entry and exit points and identifies two complete trading cycles. The first trade is initiated in January 2018 and exited in June 2019, while the second trade begins in August 2022 and is closed in March 2024. To compute the return for the two trading periods, we evaluate the profit derived from both Bitcoin and Ethereum positions, starting with an initial investment of $1000. Using the first trade as example, the calculation involves the following steps

- First Trade:

- Entry: On January 7, 2018, an amount of $1000 is used to purchase Bitcoin at a price of $16,477.60 per unit. Simultaneously, $682.213 is utilized to short Ethereum at a price of $1,153.17 per unit.
- Exit: The positions are unwound on June 28, 2019.

Table 6.15: Trading Strategy Details - Bitcoin Long and Ethereum Short

Date	**Action**	**Price Bitcoin**	**Price Ethereum**	**Long Units Bitcoin**	**Short Units Ethereum**	**Return**
2018-01-07	Entry	16,477.60	1,153.17	0.0607	0.5916	
2019-06-28	Exit	12,407.33	311.2261			251.0739
2022-08-19	Entry	20,877.55	1,612.9873	0.0479	0.4230	
2024-03-04	Exit	68,330.41	3,630.4338			1419.634

This approach ensures a systematic evaluation of trading strategy performance over the specified periods, reflecting the impact of market fluctuations on invested capital. The trading result is shown in Table 6.15. The total return of these two tradings is $1670.71.

6.4.2 Pair Trading with Ethereum and Bitcoin with Other Choices of Δ

Based on the previous regression analysis, two spread series were constructed using logarithmic prices of Bitcoin and Ethereum. These spreads are derived from the residuals of fitted linear relationships and are used to identify trading signals.

1. The first spread series with $\hat{\beta}_0 = -5.91616$ and $\hat{\beta}_1 = 1.289980$ is defined as,

$$w_t = p_{1t} - 1.289980\, p_{2t}$$

where p_{1t} and p_{2t} denote the logarithmic prices of Ethereum and Bitcoin at time t, respectively. The standard error of the residuals from this model is approximately $\hat{\sigma}_\varepsilon = 0.3995$. To define the trading threshold, we select $\Delta = 0.4$, which is slightly greater than the standard error, ensuring sensitivity to meaningful deviations while avoiding excessive noise.

2. The second spread series with $\hat{\beta}_0^* = 5.209722$ and $\hat{\beta}_1^* = 0.682213$ is defined as

$$w_t^* = p_{2t} - 0.682213\, p_{1t}$$

The standard error of the residuals in this case is $\hat{\sigma}_\varepsilon^* = 0.2905$. Accordingly, we choose $\Delta^* = 0.3$, slightly exceeding the standard error to balance responsiveness and robustness in signal detection.

The choices of thresholds Δ are critical for determining entry and exit points in the trading strategy, as they define the bounds for significant deviations from the equilibrium relationship implied by the regression models. To assess the sensitivity and robustness of the pair trading strategy, we explore alternative threshold configurations by modifying the spread deviation parameter. Specifically, we implement strategies using fractional thresholds, namely $\frac{1}{2}\Delta$ and $\frac{3}{4}\Delta$, to evaluate their impact on trading frequency and profitability.

We begin by testing the strategy with $\frac{1}{2}\Delta$. For the first spread series, after incorporating the out-of-sample dataset, the resulting spread series over time is presented in Figure 6.14, providing a clear framework for identifying potential trading signals under

Figure 6.14: Spread Series of Ethereum - Bitcoin for $\frac{1}{2}\Delta$

the reduced threshold configuration. The trading strategy is implemented using the same structure as previously described, with modified entry and exit conditions

- Entry Condition: At time t, enter a position by buying one dollar of Ethereum and shorting 1.289980 dollars of Bitcoin if $w_t = \mu_w - \frac{1}{2}\Delta$
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i} = \mu_w + \frac{1}{2}\Delta$

The program identifies all such entry and exit points by scanning the spread series and checking for values that match the entry and exit thresholds within a small numerical tolerance. The Table 6.16 outlines key trading actions involving two such trading cycles. The first trade was initiated on May 9, 2019, and closed on May 12, 2021, yielding a actual gain of $12,253.33. The second trade was entered on January 8, 2024, under similar

Table 6.16: Trading Strategy Details - Ethereum Long and Bitcoin Short with $\frac{1}{2}\Delta$

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t	**Return**
2019-05-09	Entry	6,174.53	170.29	-6.122	
2021-05-12	Exit	49,150.53	3,785.85	-5.696	12,253.33
2024-01-08	Entry	46,970.50	2,333.03	-6.122	

spread conditions. This concise summary provides an insightful snapshot of the trading strategy's performance. Comparing to the One-Delta strategy in Table 6.14, the half-delta strategy appears to be more responsive and profitable in the observed periods, capturing mean-reverting opportunities more aggressively.

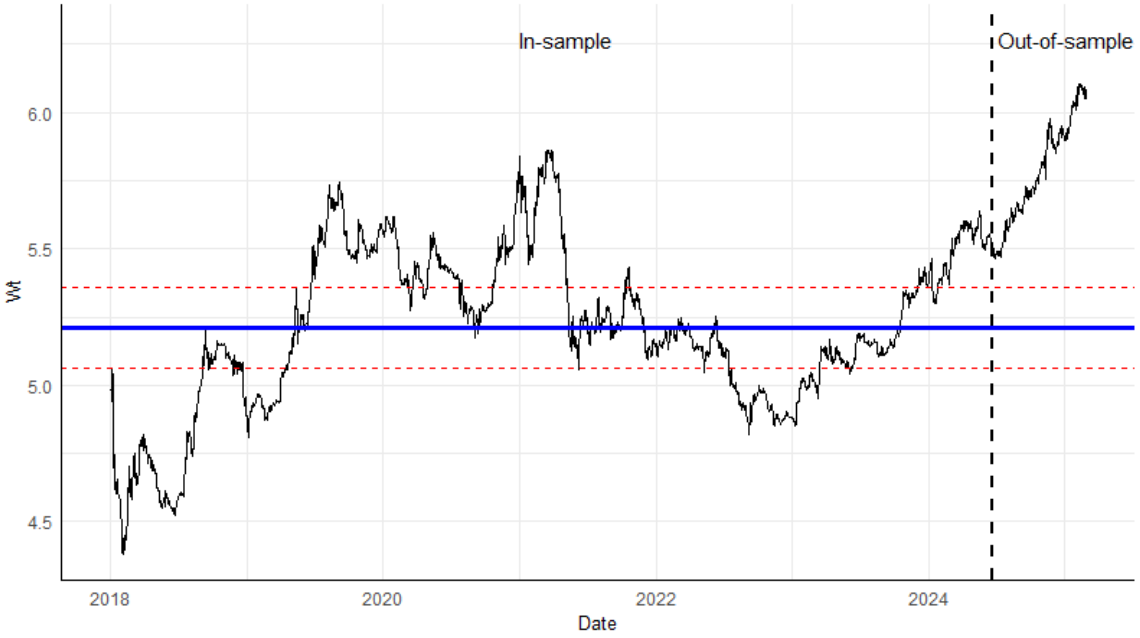


Figure 6.15: Spread Series of Bitcoin - Ethereum for $\frac{1}{2}\Delta^*$

For the second spread series, we also apply the trading strategy using a half-delta threshold. The time series of the spread, including the out-of-sample period, is illustrated in Figure 6.15. We apply a similar trading strategy as before, defined as follows

- Entry Condition: At time t, enter a position by buying one dollar of Bitcoin and shorting 0.682213 dollars of Ethereum if $w_t^* = \mu_w^* - \frac{1}{2}\Delta^*$
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i}^* = \mu_w^* + \frac{1}{2}\Delta^*$

Table 6.17: Trading Strategy Details - Bitcoin Long and Ethereum Short with $\frac{1}{2}\Delta^*$

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t	**Return**
2018-01-01	Entry	13,657.20	772.64	4.98543	
2019-05-13	Exit	7,814.92	196.85	5.36005	80.62
2021-06-07	Entry	33,560.71	2,590.26	5.05925	
2021-10-11	Exit	57,484.79	3,545.35	5.38328	461.31
2022-05-09	Entry	30,296.95	2,245.43	5.05440	
2023-12-04	Exit	41,980.10	2,243.22	5.38122	386.29

We implement the program to systematically identify the entry and exit points based on the specified trading strategy, with the results presented in Table 6.17. The table documents multiple entry and exit points, with each trade triggered by the spread crossing predefined thresholds, resulting in a total monetary gain of $928.22. While the half-delta strategy offers more trading opportunities and consistent gains, the one-delta strategy demonstrates the potential for larger individual profits. This comparison highlights the

trade-off between signal granularity and return magnitude in threshold-based pair trading strategies.

We next evaluate the strategy using a threshold of $\frac{3}{4}\Delta$, which offers an intermediate configuration between the full and half-delta approaches. The Figure 6.16 displays the spread series over time for the first spread specification, including the out-of-sample dataset, under the $\frac{3}{4}\Delta$ threshold condition.

Figure 6.16: Spread Series of Ethereum - Bitcoin for $\frac{3}{4}\Delta$

The trading strategy follows the same structure as previously described, with the following entry and exit conditions

- Entry Condition: At time t, enter a position by buying one dollar of Ethereum and shorting 1.289980 dollars of Bitcoin if $w_t = \mu_w - \frac{3}{4}\Delta$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i} = \mu_w + \frac{3}{4}\Delta$.

Table 6.18: Trading Strategy Details – Ethereum Long and Bitcoin Short with $\frac{3}{4}\Delta$

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t	**Return**
2019-05-13	Entry	7,814.91	196.85	-6.28069	
2021-06-07	Exit	33,560.71	2,590.26	-5.58351	7,909.05
2024-03-27	Entry	69,455.34	3,500.12	-6.22071	

The trading strategy based on the $\frac{3}{4}\Delta$ threshold is summarized in Table 6.18. The strategy triggered one completed trade, yielding a substantial return of \$7,909.05. In comparison, the half-delta strategy resulted in one completed trade with a higher return of \$12,253.32, while the one-delta strategy produced a return of \$3,187.20. The $\frac{3}{4}\Delta$ threshold appears to strike a balance between trade frequency and profitability, offering a compelling alternative for traders seeking fewer but more impactful positions.

For the second spread series, we apply the trading strategy using a threshold of $\frac{3}{4}\Delta^*$. The evolution of the spread over time, incorporating both in-sample and out-of-sample data, is illustrated in Figure 6.17, providing a clear framework for identifying potential trading signals. We apply a similar trading strategy as before, defined as follows

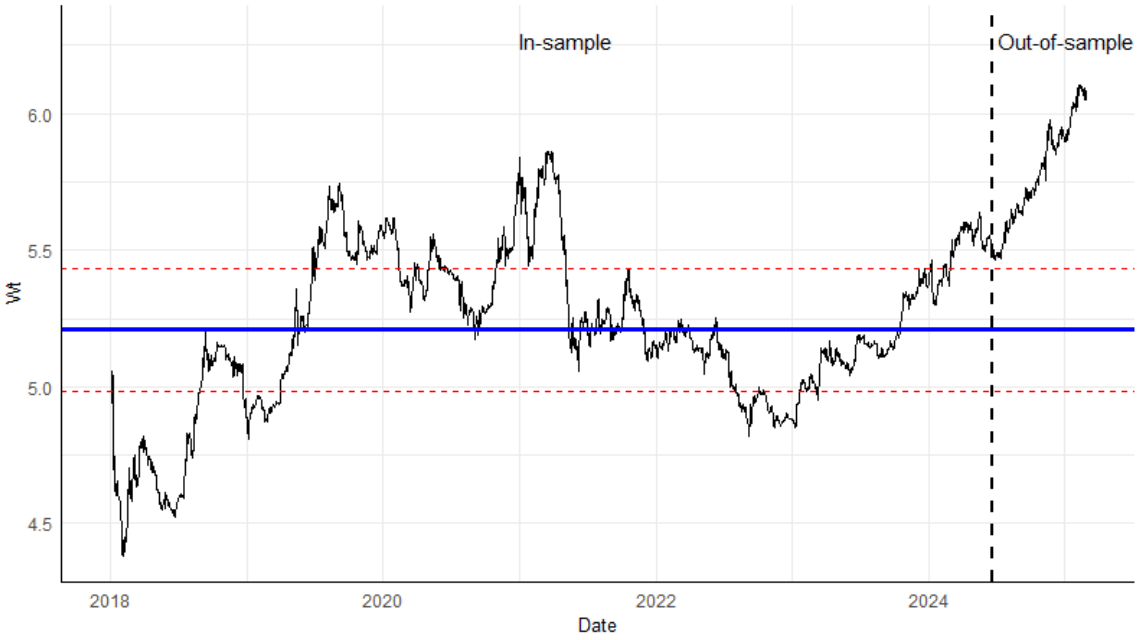


Figure 6.17: Spread Series of Bitcoin - Ethereum for $\frac{3}{4}\Delta^*$

- Entry Condition: At time t, enter a position by buying one dollar of Bitcoin and shorting 0.682213 dollars of Ethereum if $w_t^* = \mu_w^* - \frac{3}{4}\Delta^*$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i}^* = \mu_w^* + \frac{3}{4}\Delta^*$.

Table 6.19: Trading Strategy Details – Bitcoin Long and Ethereum Short with $\frac{3}{4}\Delta^*$

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t^*	**Return**
2018-01-03	Entry	15,201.00	962.72	4.94247	
2019-06-25	Exit	11,790.92	318.13	5.44387	232.45
2022-08-05	Entry	23,289.31	1,732.25	4.96837	
2024-01-08	Exit	46,970.50	2,333.03	5.46677	780.22

The program is applied to systematically identify the entry and exit points based on this strategy. The trading strategy based on the $\frac{3}{4}\Delta^*$ threshold, as presented in Table 6.19, identifies fewer but more selective trading opportunities. This approach resulted in two completed trades: the first from January 3, 2018 to June 25, 2019, yielding a return of $232.45, and the second from August 5, 2022 to January 8, 2024, yielding $780.22. In comparison, the half-delta strategy (Table 6.17) produced three trades with a total gain of $928.22, while the one-delta strategy (Table 6.15) yielded two trades totaling $1,670.71. These results suggest that the $\frac{3}{4}\Delta$ strategy offers a balanced compromise between trade frequency and profitability, making it a viable option for traders seeking moderate activity with consistent returns.

6.4.3 Pair Trading with Ethereum and Ethereum Classic

Based on the regression and Augmented Dickey–Fuller test results presented in Table 6.3 and Table 6.4, the first fitted spread series with $\hat{\beta}_0 = 3.27135$ and $\hat{\beta}_1 = 1.27403$ for Ethereum and Ethereum Classic is defined as

$$w_t = p_{1t} - 1.27403\, p_{2t}$$

where p_{1t} and p_{2t} denote the log prices of Ethereum and Ethereum Classic at time t, respectively.

The spread series w_t is found to be stationary, with a mean of $\mu_w = \hat{\beta}_0 = 3.27135$. The standard error of the residuals is approximately $\hat{\sigma}_\varepsilon = 0.5456$. Accordingly, we select the threshold parameter $\Delta = 0.55$, which slightly exceeds one standard error, ensuring that trading signals are triggered by statistically meaningful deviations from the equilibrium relationship.

Figure 6.18: Spread Series Over Time for Ethereum - Ethereum Classic

To implement the trading strategy, we incorporate out-of-sample data from June 18, 2024, to February 27, 2025. This new dataset is merged with the original dataset used to develop the strategy. The spread series for the extended dataset is then computed using the same formula as above. In Figure 6.18, we plot the spread series between the daily log prices of Ethereum and Ethereum Classic, including horizontal lines representing the mean μ_w, upper threshold $\mu_w + \Delta$, and lower threshold $\mu_w - \Delta$. The trading strategy is defined as follows

- Entry Condition: At time t, enter a position by buying one dollar of Ethereum and shorting 1.27403 dollars of Ethereum Classic if $w_t = \mu_w - \Delta$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i} = \mu_w + \Delta$.

Table 6.20: Trading Strategy Details - Ethereum Long and Ethereum Classic Short

Date	**Action**	**Ethereum Price**	**ETC Price**	w_t	**Return**
2018-01-01	Entry	772.64	34.17	2.15085	
2020-09-04	Exit	388.24	5.36	3.82176	576.53
2021-05-04	Entry	3253.63	69.49	2.68412	
2022-05-11	Exit	2072.11	19.78	3.83395	548.30
2022-09-05	Entry	1617.18	39.51	2.70452	
2023-05-24	Exit	1800.10	17.79	3.82820	813.47

As summarized in Table 6.20, the approach involves three distinct trading periods: from January 1, 2018 to September 4, 2020; from May 4, 2021 to May 11, 2022; and from September 5, 2022 to May 24, 2023. Each round-trip trade yielded substantial profits, with a cumulative actual return of $1,938.30, demonstrating the effectiveness of the mean-reversion-based trading strategy between Ethereum and Ethereum Classic.

Based on the reversed-order regression model presented in Table 6.9, the second estimated spread series for the pair Ethereum Classic - Ethereum is defined as

$$w_t^* = p_{2t} - 0.609331\, p_{1t}$$

where where p_{1t} and p_{2t} denote the log prices of Ethereum and Ethereum Classic at time t, respectively.

The spread series w_t^* has a standard error of approximately 0.3773. To define a threshold band for identifying statistically meaningful deviations from the equilibrium, we select $\Delta^* = 0.4$, which is slightly greater than the standard error. After incorporating the out-of-sample period data, we compute and plot the spread series in Figure 6.19, which

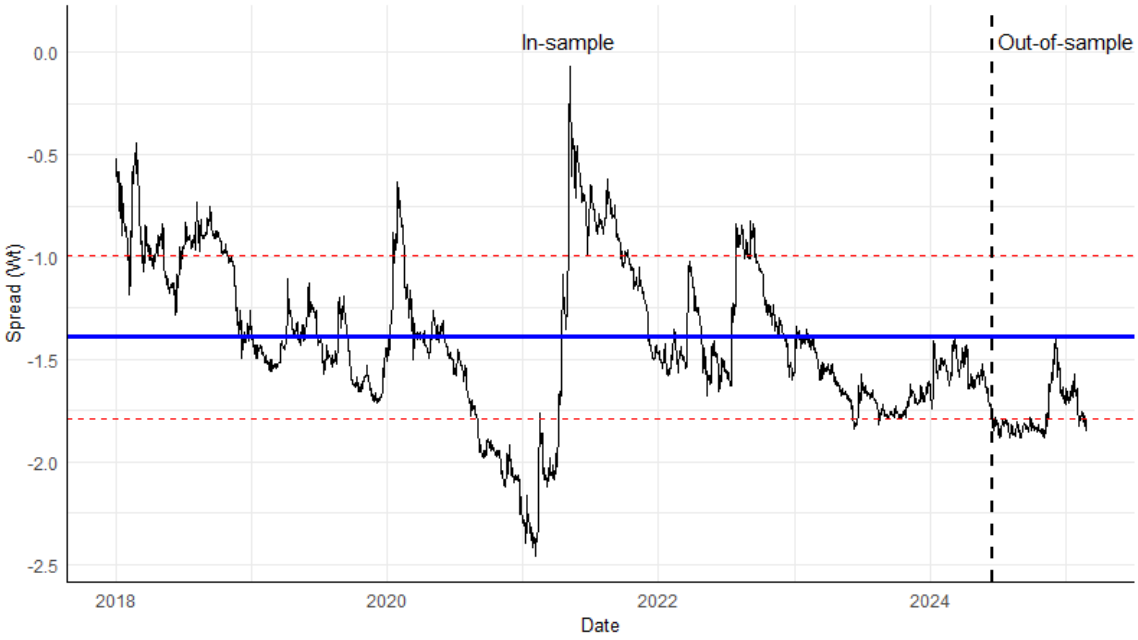


Figure 6.19: Spread Series Over Time for Ethereum Classic - Ethereum

includes horizontal lines representing the mean of the spread, as well as the upper and lower bounds defined by $\mu_w^* \pm \Delta^*$. The trading strategy is defined as

- Entry Condition: At time t, initiate a position by buying one dollar of Ethereum Classic and shorting 0.609331 dollars of Ethereum if $w_t^* = \mu_w^* - \Delta^*$.
- Exit Condition: Close the position at time $t+i$ (for some $i > 0$) when $w_{t+i}^* = \mu_w^* + \Delta^*$.

The trading strategy is implemented and Table 6.21 summarizes the execution details. The results reveal one completed trade, initiated on 2020-08-30 and closed on 2021-05-04, which yielded a actual return of $5375.92. This outcome highlights the strategy's potential to generate substantial profits by exploiting temporary deviations in the spread from its long-run equilibrium.

Table 6.21: Trading Strategy Details – Ethereum Classic Long and Ethereum Short

Date	**Action**	**Ethereum Price**	**ETC Price**	w_t^*	**Return**
2020-08-30	Entry	428.40	6.69	-1.7926	
2021-05-04	Exit	3253.63	69.49	-0.6868	5375.92
2023-06-10	Entry	1752.38	15.07	-1.8386	

6.4.4 Pair Trading with Bitcoin and Ethereum Classic

The regression analysis using the logarithm of Bitcoin price as the dependent variable and the logarithm of Ethereum Classic price as the independent variable is presented in Table 6.5, with the corresponding cointegration test results shown in Table 6.6. Although the ADF test does not support the presence of cointegration between the two series, we proceed to apply a similar pairs trading strategy to explore potential trading opportunities. Based on the regression results, the spread series is defined as

$$w_t = p_{1t} - 0.78453\, p_{2t}$$

where p_{1t} and p_{2t} denote the log prices of Bitcoin and Ethereum Classic at time t, respectively.

The standard error of the spread series is approximately 0.5586. We select a threshold of $\Delta = 0.56$, which is slightly greater than one standard deviation, to define the entry and exit points for the trading strategy. Incorporating out-of-sample data, the spread series is plotted in Figure 6.20, with horizontal lines indicating the mean and the upper and lower thresholds defined by $\mu_w \pm \Delta$. The trading strategy is defined as

- Entry Condition: At time t, initiate a position by buying one dollar of Bitcoin and shorting 0.78453 dollars of Ethereum Classic if $w_t = \mu_w - \Delta$.

Figure 6.20: Spread Series Over Time for Bitcoin - Ethereum Classic

- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i} = \mu_w + \Delta$.

Table 6.22: Trading Strategy Details – Bitcoin Long and Ethereum Classic Short

Date	**Action**	**Bitcoin Price**	**ETC Price**	w_t	**Return**
2018-01-01	Entry	13657.20	34.17	6.7516	
2020-11-01	Exit	13737.11	5.21	8.2322	670.64
2021-05-06	Entry	56396.52	134.10	7.0971	
2023-10-24	Exit	33901.53	16.45	8.2341	289.40

The Table 6.22 summarizes the execution of the trading strategy. During the analysis period, two complete trades were executed, with a total gain of $960.04. The results suggest that the strategy is capable of capturing profitable deviations in the spread between Bitcoin and Ethereum Classic, even in the absence of formal cointegration.

Figure 6.21: Spread Series Over Time for Ethereum Classic - Bitcoin

From the regression results with the reversed order, as shown in Table 6.11, the second estimated spread series is defined as

$$w_t^* = p_{2t} - 0.70949\, p_{1t}$$

where p_{1t} and p_{2t} denote the log prices of Bitcoin and Ethereum Classic at time t, respectively. The standard error of the spread series is approximately 0.5312. We then select a threshold of $\Delta^* = 0.54$, which is slightly greater than one standard deviation. To implement the trading strategy, we incorporate out-of-sample period data and visualize the spread series in Figure 6.21. The trading strategy is defined as follows

- Entry Condition: At time t, initiate a position by buying one dollar of Ethereum Classic and shorting 0.70949 dollar of Bitcoin if the spread $w_t^* = \mu_w^* - \Delta^*$.
- Exit Condition: Unwind the position at time $t+i$ (for some $i > 0$) when $w_{t+i}^* = \mu_w^* + \Delta^*$.

Table 6.23: Trading Strategy Details – Ethereum Classic Long and Bitcoin Short

Date	**Action**	**Bitcoin Price**	**ETC Price**	w_t^*	**Return**
2019-07-14	Entry	10256.06	5.67	-4.8173	
2021-05-04	Exit	53333.54	69.49	-3.4811	8274.82
2024-08-01*	Entry	65409.83	21.64	-4.7927	

* Occurs during the Out-of-sample period.

The Table 6.23 presents the implementation details of a trading strategy. A complete trade was initiated on July 14, 2019 and closed on May 4, 2021, yielding a return of $8274.82. This table illustrates how the spread threshold effectively identifies entry and exit points for profitable trades.

6.5 Implementation of Pair Trading with Training and Test Data

To evaluate the robustness and out-of-sample performance of the pair trading strategy, we employ a two-phase framework using separate training and test datasets. The training dataset is used to construct the strategy by identifying cointegrated pairs, estimating regression parameters, defining the trading signals, and calibrating the spread threshold. Once the strategy is built, it will be applied to a separate test dataset to evaluate its performance. The test phase will assess whether the strategy remains effective out-of-sample and will estimate the expected return based on historical data. The training dataset consists of the logarithmic prices of Bitcoin and Ethereum covering the period from January 1, 2018 to December 31, 2021. This period contains a total of 1461 days. The testing dataset spans January 1, 2022, through February 27, 2025, providing an additional 1154 daily observations for out-of-sample evaluation.

Table 6.24: Linear Regression of Bitcoin on Ethereum - Training Dataset

Variable	**Estimate**	**Std. Error**	**t value**	**Pr(>|t|)**
$\hat{\beta}_0$	5.22423	0.04810	108.61	$< 2 \times 10^{-16}$
$\hat{\beta}_1$ (Ethereum)	0.68358	0.00769	88.89	$< 2 \times 10^{-16}$
Model Statistics				
Residual standard error: 0.3321 on 1459 degrees of freedom				
Multiple R-squared: 0.8441				
Adjusted R-squared: 0.844				
F-statistic: 7901 on 1 and 1459 DF				
p-value: $< 2.2 \times 10^{-16}$				

We perform a linear regression on the training dataset, and the results are presented in Table 6.24. Based on the estimated coefficients, the fitted spread series is defined as

$$w_t = p_{1t} - 0.68358\, p_{2t}$$

Figure 6.22: Spread Series for Bitcoin - Ethereum with Training Dataset

where p_{1t} and p_{2t} denote the log prices of Bitcoin and Ethereum at time t, respectively. The standard deviation of the spread series w_t is approximately 0.3319. Accordingly, we select a threshold of $\Delta = 0.35$, which is slightly greater than one standard deviation and helps account for typical fluctuations in the spread.

Figure 6.23: Spread Series for Bitcoin - Ethereum with Training and Testing Datasets

The time series plot of the spread during the training period is shown in Figure 6.22. Once the strategy is constructed, we apply it to an out-of-sample test dataset to evaluate its performance beyond the training window. Using the previously estimated regression parameters $\hat{\beta}_0 = 5.22423$ and $\hat{\beta}_1 = 0.68358$, we recompute the spread series over the full dataset and generate trading signals based on the defined thresholds. The evolution of the spread over time, including both in-sample and out-of-sample periods, is visualized in Figure 6.23. Following the same spread-based trading framework described in the previous section, we define the trading strategy as follows

- Entry Condition: At time t, initiate a position by buying one dollar of Bitcoin and shorting 0.68358 dollar of Ethereum if the spread $w_t = \mu_w - \Delta$.

- Exit Condition: Unwind the position at time $t+i$ (for some $i>0$) when $w_{t+i} = \mu_w + \Delta$.

Table 6.25: Trading Strategy Details – Bitcoin Long and Ethereum Short for Training and Testing Datasets

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	**Return**
2018-01-08	Entry	15,170.10	1,148.53	
2019-07-15	Exit	10,895.09	229.78	265.0203
2022-09-01*	Entry	20,127.14	1,586.18	
2024-03-28*	Exit	70,744.95	3,561.29	1,663.7006

* Occurs during the Out-of-sample period.

The trading outcomes are summarized in Table 6.25. Across two completed pair trades, the strategy yielded a cumulative return of $1928.72. This result demonstrates the effectiveness of the spread-based framework in capturing profitable opportunities under both in-sample and out-of-sample conditions.

6.6 Pair Trading with Orthogonal Distance Regression Models

In traditional regression analysis, ordinary least squares (OLS) assumes that only the dependent variable is subject to measurement error, while it is conditional on the values of independent variable, which are held fixed. Thus, functionally it minimizes the sum of squared vertical distances between observed and predicted values of the dependent values. However, in financial contexts such as here for cryptocurrencies, both asset prices are subject to noise. Orthogonal distance regression (ODR) addresses this limitation by minimizing the perpendicular (orthogonal) distances of the data points from the fitted line,

see Figures 6.24 and 6.25, providing a more balanced estimation when both variables exhibit random fluctuations. Thus the roles of dependent and independent variables are eliminated all together, allowing the same equation to express either variable in terms of the other.

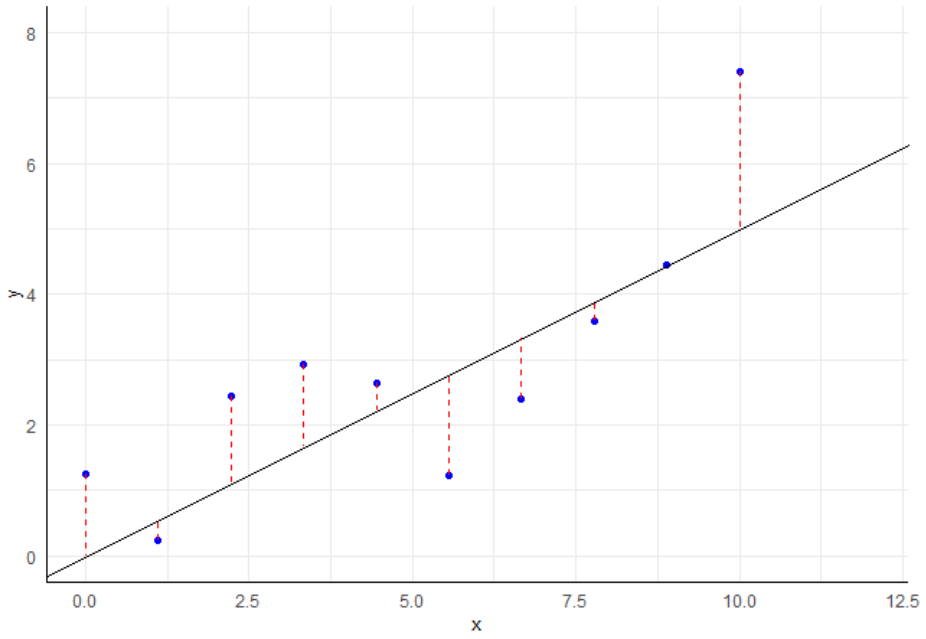


Figure 6.24: OLS Model with Vertical Distances

We apply ODR model to the cryptocurrency pair of Ethereum and Bitcoin. In the initial OLS model, where the logarithm of Bitcoin price is treated as the independent variable and the logarithm of Ethereum price as the dependent variable, the fitted relationship is

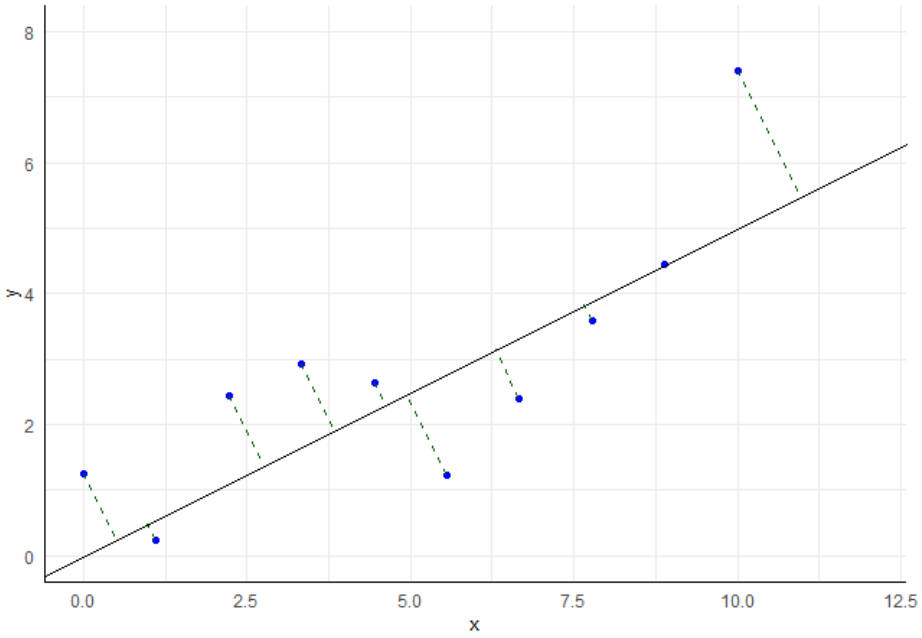


Figure 6.25: ODR Model with Perpendicular Distances

$$\hat{p}_{1t} = -5.916 + 1.290\, p_{2t}$$

where p_{1t} and p_{2t} denote the logarithmic prices of Ethereum and Bitcoin at time t, respectively. The regression results are presented in Table 6.1, with an R^2 value of 0.88. With an interchange of their respective roles in the reverse, we also have

$$\hat{p}_{2t} = 5.210 + 0.682\, p_{1t}$$

In contrast, when we perform the orthogonal distance regression on the same dataset to account for measurement errors in both variables. The fitted model is

$$\hat{p}^{*}_{1t} = -7.024 + 1.403\, p_{2t}$$

which can alternatively be expressed also as

$$\hat{p}_{2t} = \frac{7.024}{1.403} + \frac{1}{1.403}\, p_{1t} = 5.006 + 0.713\, p_{1t}$$

The sum of squared orthogonal distances from the observed points to the regression line is SSQ $= 133.9259$. Using this, we define the coefficient of determination based on orthogonal residuals as

$$R^2_{ortho} = 1 - \frac{\text{SSQ}}{\text{SST}} = 0.9573$$

where SST $= \sum_i (y_i - \bar{y})^2$ is the total sum of squares. Using the traditional vertical residuals from the OLS model, where SSE $= \sum_i (y_i - \hat{y})^2$, the coefficient of determination is

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = 0.8733$$

However, two are not comparable.

To visually compare the two regression methods, the Figure 6.26 presents a scatter plot of log-transformed Bitcoin and Ethereum prices, overlaid with the fitted lines from both OLS and ODR. The OLS line minimizes vertical deviations and is optimized for predicting Ethereum log prices from Bitcoin log prices under the assumption that only Ethereum prices contain measurement error. In contrast, the ODR line accounts for errors in both variables by minimizing the orthogonal distances from the data points to the regression line.

Based on the first estimated ODR model, we compute the spread series as $w_t = p_{1t} - 1.403245\, p_{2t}$, and select a threshold of $\Delta = 0.42$, slightly above the standard deviation of the spread $\sigma_w = \hat{\sigma}_\varepsilon = 0.4106$. The time series plot of the spread is shown in

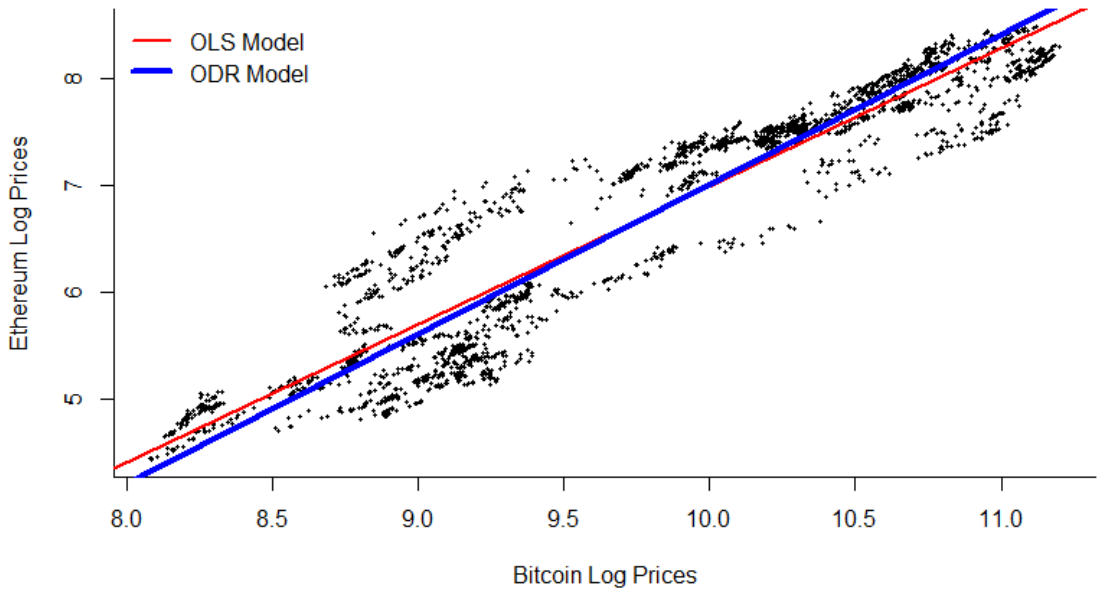


Figure 6.26: Comparison of OLS and ODR Models for Regression of Ethereum on Bitcoin

Figure 6.27, with horizontal reference lines at the μ_w and $\mu_w \pm \Delta$. The trading strategy is defined as follows

- Entry Condition: At time t, initiate a position by buying one dollar of Ethereum and shorting 1.403245 dollars of Bitcoin if $w_t = \mu_w - \Delta$.
- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w_{t+i} = \mu_w + \Delta$.

The trading outcomes based on this ODR model are summarized in Table 6.26. To further explore trading opportunities within this pair, we also construct a strategy using the reversed regression model $\hat{p}_{2t} = 5.006 + 0.713\, p_{1t}$, from which we compute the corresponding spread series as $w_t^* = p_{2t} - 0.7126339\, p_{1t}$ and select a threshold of $\Delta^* = 0.30$ that is slightly greater than the standard deviation of the spread $\sigma_w^* = 0.2926$.

Figure 6.27: Spread Series for Ethereum - Bitcoin with ODR Model

Table 6.26: Trading Strategy Details – Ethereum Long and Bitcoin Short on ODR Model

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t	**Return**
2019-06-26	Entry	13016.23	336.7532	-7.474926	
2022-08-12	Exit	24402.82	1957.2465	-6.596924	3584.553
2024-03-20	Entry	67913.67	3513.3931	-7.448156	

The Figure 6.28 displays the spread series over time, computed from the reversed ODR model. The corresponding trading strategy is defined as follows

- Entry Condition: At time t, initiate a position by buying one dollar of Bitcoin and shorting 0.7126339 dollars of Ethereum if $w_t^* = \mu_w^* - \Delta^*$

Figure 6.28: Spread Series for Bitcoin - Ethereum with ODR Model

- Exit Condition: Unwind the position at time $t+i$ (where $i > 0$) if $w^*_{t+i} = \mu^*_w + \Delta^*$

Table 6.27: Trading Strategy Details – Bitcoin Long and Ethereum Short on ODR Model

Date	**Action**	**Bitcoin Price**	**Ethereum Price**	w_t	**Return**
2018-01-07	Entry	16477.60	1153.170	4.685496	
2019-06-26	Exit	13016.23	336.7532	5.326886	294.4627
2022-08-12	Entry	24402.82	1957.2465	4.701192	
2024-03-20	Exit	67913.67	3513.3931	5.307809	1216.4324

Based on the trading details presented in Table 6.26 and 6.27, a total of three complete trades were executed using the ODR model on the Bitcoin–Ethereum pair. Collectively, these trades demonstrate the effectiveness of the ODR model in capturing profitable opportunities through mean-reverting spread series between the two assets. To evaluate the relative performance of the two modeling approaches, we compare the cumulative returns from the ODR-based strategy with those obtained from the OLS-based strategy for the same pair, as detailed in Table 6.14 and Table 6.15. The total cumulative return from the ODR-based strategy is $5095.45, while the OLS-based strategy produced a total return of $4857.91. Although both strategies demonstrate profitability, the ODR-based approach slightly outperformed its OLS counterpart in this case. This result highlights the practical advantage of orthogonal distance regression in capturing more balanced relationships when both assets are subject to measurement error. While OLS remains a widely used and computationally efficient method, ODR offers a more symmetric treatment of variables, which can lead to improved trading performance in noisy financial environments.

6.7 Concluding Remarks

This chapter explores the possibility and effectiveness of cointegration-based pair trading strategies among cryptocurrencies, with a particular focus on Bitcoin, Ethereum, and Ethereum Classic. The results confirm the presence of long-run equilibrium relationships between these assets, as assessed by the cointegration analyses, and demonstrate that such relationships can be leveraged to design profitable trading strategies.

Trading strategies based on spread thresholds are implemented and evaluated across both in-sample and out-of-sample periods. A summary of cointegration results and trading performance for various pairs and threshold configurations is presented in Table

Table 6.28: Cointegration Results and Trading Details for Various Spread Pairs

Spread Pair	**P-value**	**Cointegration Result**	**Return**
Ethereum - Bitcoin	0.0721	Marginally cointegrated	3187.20
Bitcoin - Ethereum	0.0856	Marginally cointegrated	1670.71
Total			**4857.91**
Ethereum - Bitcoin with $\frac{1}{2}\Delta$		Marginally cointegrated	12253.33
Bitcoin - Ethereum with $\frac{1}{2}\Delta$		Marginally cointegrated	928.22
Total			**13181.55**
Ethereum - Bitcoin with $\frac{3}{4}\Delta$		Marginally cointegrated	7909.05
Bitcoin - Ethereum with $\frac{3}{4}\Delta$		Marginally cointegrated	1012.67
Total			**8921.72**
Ethereum - Bitcoin with ORD		Marginally cointegrated	3584.55
Bitcoin - Ethereum with ORD		Marginally cointegrated	1510.90
Total			**5095.45**
Ethereum - Ethereum Classic	0.0137	Cointegrated	1938.30
Ethereum Classic - Ethereum	0.0021	Cointegrated	5375.92
Total			**7314.22**
Bitcoin - Ethereum Classic	0.1290	Not cointegrated	960.04
Ethereum Classic - Bitcoin	0.0145	Cointegrated	8274.82
Total			**9234.86**

6.28. The Ethereum - Bitcoin pair exhibited marginal cointegration, yielding one complete trade with a return of $3187.20, and an additional entry signal during the out-of-sample period. Its reverse specification, Bitcoin - Ethereum, also showed marginal cointegration and produced two completed trades. The Ethereum - Ethereum Classic pair demonstrated strong cointegration, generating three trades but with moderate individual returns. The reversed Ethereum Classic - Ethereum pair exhibited significant cointegration as well and produced a single trade with a substantial gain of $5375.92, alongside an additional entry signal that remained open at the end of the analysis period. Notably, the Bitcoin - Ethereum Classic pair failed to meet formal cointegration test, yet it still delivered two profitable trades with a total return of $960.04. In contrast, its reverse specification,

Ethereum Classic - Bitcoin, exhibited significant cointegration and generated one completed trade yielding $8274.82 along with an additional entry signal during the out-of-sample period.

The strength of cointegration plays a pivotal role in determining the stability of the spread and the reliability of trading signals. Pairs exhibiting strong cointegration, such as Ethereum - Ethereum Classic and Ethereum Classic - Bitcoin, generated consistent and statistically robust opportunities for mean-reversion trading. Conversely, pairs with marginal or absent cointegration, such as Bitcoin - Ethereum Classic, still produced profitable trades, though with greater variability and less predictability. This finding suggests that while cointegration enhances robustness, its absence does not entirely preclude profitability, particularly in highly volatile markets such as cryptocurrencies. The magnitude of return is primarily influenced by the standard deviation of the spread series and the choice of trading thresholds.

The analysis revealed that reversing the roles of dependent and independent variables in regression models can significantly affect cointegration test outcomes. For instance, the Bitcoin - Ethereum Classic pair was not cointegrated in one direction but exhibited strong stationarity when reversed. This asymmetry reflects the directional nature of ordinary least squares estimation and highlights the practical advantage of considering both regression orientations when constructing trading strategies. Such flexibility enables traders to exploit whichever specification yields a more stable spread.

The selection of spread thresholds (Δ, $\frac{1}{2}\Delta$, and $\frac{3}{4}\Delta$) significantly influenced trading frequency and profitability. Narrower thresholds increased trade frequency but may reduce individual profitability, and introduced potentially higher transaction costs. Wider thresholds produced fewer but more conservative trades, while intermediate thresholds offered a balanced compromise between trade frequency and return magnitude. For the

pair between Ethereum and Bitcoin, the half-delta strategy with one complete trade appeared to be more profitable comparing to the one-delta strategy in the Ethereum - Bitcoin pair. Whereas for the reversed Bitcoin - Ethereum pair, he half-delta strategy offers more trading opportunities, but less individual profits than the one-delta strategy, highlights the trade-off between signal granularity and return magnitude in threshold-based pair trading strategies. The $\frac{3}{4}\Delta$ threshold appears to strike a balance between trade frequency and profitability. These findings emphasize the importance of calibrating thresholds to align with market conditions and risk preferences.

The comparison between ordinary least squares and orthogonal distance regression demonstrated that ODR provides a more symmetric and robust estimation of hedge ratios by accounting for measurement errors in both assets. The ODR-based strategies slightly outperformed their OLS counterparts in cumulative returns for the pairs between Bitcoin and Ethereum, underscoring its practical advantage in noisy financial environments. While OLS remains computationally efficient, ODR offers improved precision for markets characterized by bidirectional volatility.

The evaluation of in-sample and out-of-sample performance confirmed that strategies calibrated on historical data retained profitability in unseen periods, indicating reasonable generalizability. However, variations in returns across pairs and thresholds underscore the sensitivity of these strategies to evolving market dynamics, reinforcing the need for periodic recalibration and robust risk management.

The analysis demonstrates that cointegration-based pair trading can be a viable statistical strategy in cryptocurrency markets, particularly when combined with robust estimation techniques such as ODR. However, profitability remains sensitive to threshold selection, transaction costs, and market volatility.

CHAPTER SEVEN

TIME-VARYING VOLATILITY ANALYSIS FOR CRYPTOCURRENCIES

7.1 Introduction

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model is a widely used time-series framework for modeling time-varying volatility, especially for financial returns data where volatility tends to exhibit clustering over time. The crypto data appears to have such a clustering process. Thus it is appropriate to explore this aspect of the crypto-analysis.

Chu et al. (2017) have conducted a broad application of Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models to multiple cryptocurrencies, demonstrating that volatility across major coins is highly persistent and heavy-tailed, and showing that more flexible GARCH variants substantially improve the model fit. Almansour et al. (2021) analyzed nine major cryptocurrencies and found that both ARCH and GARCH models significantly improve volatility forecasting, with past volatility and news shocks exerting strong and persistent effects on conditional volatility across all assets. Ngunyi et al. (2019) analyzed the volatility dynamics of eight major cryptocurrencies using a broad set of GARCH-family models and showed that although no single specification dominates across assets, the asymmetric GARCH models with long memory property and heavy-tailed innovations distributions overall perform better for all cryptocurrencies. However, their data were limited only for the early years of the emergence of crypto. Here we analyze a more up-to-date and more extensive data.

The GARCH model consists of two equations in the model structure, namely a mean equation and a variance equation. The mean equation models the dynamics of the observed time series, such as returns, and is commonly specified as either a constant or as

an autoregressive (AR) process. The variance equation describes how the conditional volatility evolves over time according to the GARCH structure. The model can be written as

$$r_t = \mu + \varepsilon_t, \qquad \varepsilon_t = \sigma_t z_t \tag{7.1}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{7.2}$$

where

- r_t denotes the asset return at time t, typically measured as the percentage change in price from one period to the next.
- σ_t^2 represents the conditional volatility of returns at time t.
- $\omega > 0$ is the baseline variance.
- $\alpha \geq 0$ measures the impact of recent shocks, often referred to as the ARCH effect.
- $\beta \geq 0$ reflects the persistence of volatility, often referred to as the GARCH effect.
- $z_t \sim N(0,1)$ or standardized Student's t distribution, is a sequence of independent and identically distributed random variables with mean zero, variance equal to one.

The condition $(\alpha + \beta) < 1$ ensures the stationarity of the process. When interpreting the parameter estimates, a large value of α indicates that the conditional variance responds strongly to new shocks, whereas a large value of β implies that volatility decays slowly over time. When the sum $\alpha + \beta$ is close to one, the volatility process is highly persistent. The commonly used approach for estimating GARCH model parameters is the method of maximum likelihood. Hansen and Lunde (2005) demonstrate

that it is difficult to identify volatility models that outperform the simple GARCH(1,1) specification. Consequently, for many empirical applications the GARCH(1,1) model provides an adequate and robust fit. This model can be written as

$$\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{7.3}$$

In our analysis, we estimate the GARCH model using the the R package *rugarch*, as documented in Ghalanos(2020), Perlin et al (2020). The package provides a comprehensive suite of diagnostic tests, also described in Ghalanos(2020). In particular, it implements the weighted Ljung–Box and ARCH Lagrange Multiplier (LM) statistics of Fisher and Gallagher (2012), which better account for the distribution of the statistics computed from the estimated models. The weighted ARCH–LM test as a weighted portmanteau test evaluates the null hypothesis of an adequately specified ARCH process, while the weighted Ljung–Box test serves as another portmanteau test for assessing the adequacy of the ARMA specification.

The Sign Bias Test of Engle and Ng (1993) examines whether the standardized residuals exhibit asymmetric responses to past positive or negative shocks by regressing the squared standardized residuals on indicators of past shocks,

$$\hat{z}_t^2 = c_0 + c_1 I_{\hat{\varepsilon}_{t-1}<0} + c_2 I_{\hat{\varepsilon}_{t-1}<0}\hat{\varepsilon}_{t-1} + c_3 I_{\hat{\varepsilon}_{t-1}\geq 0}\hat{\varepsilon}_{t-1} + u_t \tag{7.4}$$

where I denotes the indicator function, and $\hat{\varepsilon}_t$ are the estimated residuals from the GARCH model. The Null Hypotheses are $H_0 : c_i = 0$ for each $i = 1,2,3$, and the joint hypothesis $H_0 : c_1 = c_2 = c_3 = 0$.

The chi-squared goodness of fit test based on Palm (1996), compares the empirical distribution of the standardized residuals with the theoretical distribution implied by the

chosen density. The test accommodates non-i.i.d. observations by reclassifying standardized residuals according to their magnitude and calculating the probability of observing a value smaller than the standardized residual, which under the null should follow a standard uniform distribution. The *rugarch* implementation requires specifying the fitted model and the number of bins, and we consider the default choices of 20, 30, 40, and 50 bins. The parameter stability and joint stability tests of Nyblom (1989) are also included in the diagnostic output, which reports the corresponding test statistics and critical values.

7.2 Univariate GARCH Model

We first begin by analyzing the daily returns of Bitcoin. As illustrated in Figure 7.1, the daily return series fluctuates around zero and does not exhibit a persistent trend, which supports specifying a zero or near-zero mean in the conditional mean equation of the GARCH models. The plot shows clear volatility clustering, where calm periods alternate with bursts of large positive and negative moves, especially around major market events in early 2020. And those bursts tend to persist for a while, that clumping of large moves justifies modeling the conditional variance dynamically in the classic GARCH model. Moreover, the frequent sign flips in the daily returns suggest a minimal autocorrelation in the mean, while the pronounced clustering of large movements strongly supports the use of GARCH type volatility models to capture Bitcoin's time varying risk dynamics.

We compute monthly volatility using a 30-day rolling window. For each date t, we collect the most recent 30 daily returns for each asset and calculate their standard deviation. To annualize this measure, we multiply the standard deviation by the square root of 365, reflecting the fact that cryptocurrency markets operate continuously

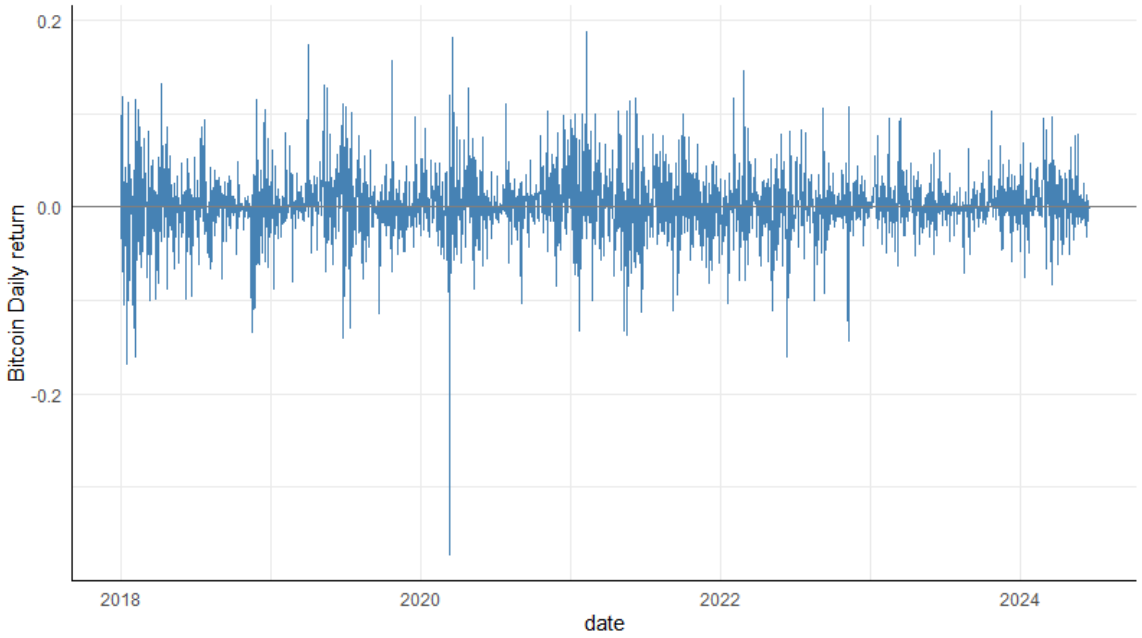


Figure 7.1: Time Series Plot of Bitcoin Daily Returns

throughout the year. The resulting values are recorded as the volatilities at time corresponding time point. Each point thus represents the variability of daily returns over the preceding month, scaled to an annual frequency, thereby illustrating how Bitcoin's short-term risk evolves over time. These are presented in Figure 7.2.

The series exhibits a highly time varying volatility; volatility spikes sharply during major market disruptions, e.g. it spikes most prominently around the early during 2020 COVID outbreak and the annualized volatility shoots above 1.6, indicating an extremely turbulent regime with frequent large daily swings. Earlier periods such as early 2018 and parts of 2019 and 2021 also show elevated volatility. After each spike, volatility gradually declines, showing the volatility clustering pattern where calm conditions follow periods of stress but only slowly. From 2022 onward, although volatility remains elevated relative to

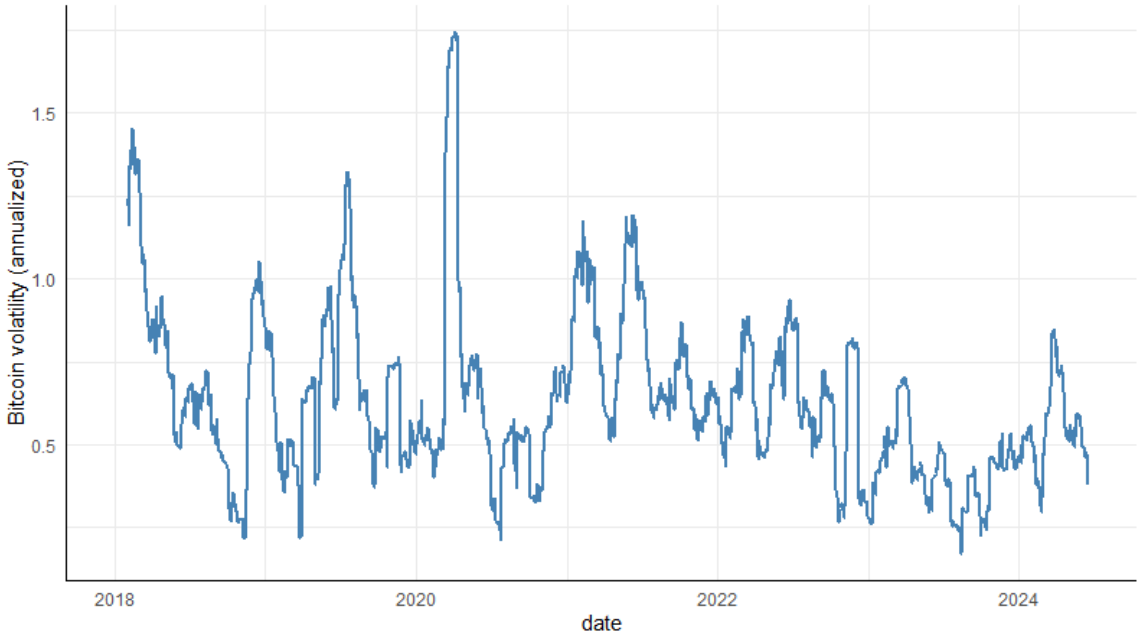


Figure 7.2: Monthly Volatility (Annualized) of Bitcoin Returns

traditional assets, it settles into a more moderate range, reflecting a somewhat more mature market structure.

We fit the GARCH(1,1) model on the Bitcoin daily returns, with constant mean and t-distribution, which is usually preferred for heavy-tailed returns. The results are presented in Table 7.1. The mean equation indicates that the estimated unconditional expected daily return $\hat{\mu} = 0.000747$ is not significantly different from zero. This is typical for daily cryptocurrency returns. This reinforces the appropriateness of modeling Bitcoin returns as a zero-mean process for risk and volatility applications.

The conditional variance parameters reveal a highly persistent volatility structure. The estimated ARCH coefficient $\hat{\alpha}_1 = 0.0732$ is positive and strongly significant, indicating that recent shocks to returns exert an immediate effect on volatility. The GARCH coefficient estimate $\hat{\beta}_1 = 0.9258$ is also highly significant and close to unity. The

Table 7.1: GARCH Model Fit for Bitcoin Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	0.000747	0.000459	1.6287	0.10337
ω	0.000012	0.000009	1.2591	0.20798
α_1	0.073212	0.010236	7.1526	0.00000
β_1	0.925788	0.015596	59.3614	0.00000
shape	3.173847	0.219148	14.4827	0.00000
Information Criteria: Akaike = -4.1697, Bayes = -4.1575, Shibata = -4.1697 Hannan–Quinn = -4.1653, Log-Likelihood = 4925.285				
Weighted Ljung-Box Test	Statistic	P value		
Lag[1]	0.6528	0.4191		
Lag[$2(p+q)+(p+q)-1$][2]	2.4371	0.1995		
Lag[$4(p+q)+(p+q)-1$][5]	5.7098	0.1052		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	1.240	0.500	2.000	0.2655
ARCH Lag[5]	2.275	1.440	1.667	0.4136
ARCH Lag[7]	2.426	2.315	1.543	0.6277
Nyblom Stability Test	Value			
Joint Statistic	4.5232			
Individual Statistics				
μ	0.08634			
ω	0.38674			
α_1	1.11146			
β_1	0.65972			
shape	0.35516			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	0.02800	0.9777		
Negative Sign Bias	0.02083	0.9834		
Positive Sign Bias	0.73083	0.4650		
Joint Effect	0.81133	0.8468		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	51.63	7.522×10^{-5}		
30	68.02	5.635×10^{-5}		
40	71.59	1.125×10^{-3}		
50	88.56	4.613×10^{-4}		

sum of these parameters ($\hat{\alpha}_1 + \hat{\beta}_1 \approx 0.999$) suggests that volatility is extremely persistent and decays very slowly over time. This implies that the volatility shocks have the long-lasting effects on Bitcoin's risk profile. The insignificance of the long-run variance constant estimate $\hat{\omega}$ is typical in near-IGARCH settings (where $\alpha + \beta = 1$), in which shocks to volatility decay very slowly or essentially persist indefinitely, and past shocks rather than the unconditional component dominate long-run variance dynamics.

The shape parameter of the Student-t distribution with estimated degree of freedom $\hat{\nu} \approx 3.17$ indicates the presence of extremely heavy tails in the return distribution. This confirms that using a heavy-tailed distribution is appropriate for capturing the extreme movements that frequently occur in Bitcoin returns.

The weighted Ljung–Box tests applied to both standardized residuals and their squares yield high p-values across multiple lags, indicating the absence of remaining serial correlation or omitted conditional heteroskedasticity, see Table 7.1. Similarly, the ARCH LM tests fail to reject the null hypothesis of no remaining ARCH effects, demonstrating that the GARCH model effectively captures the volatility clustering exhibited in the raw returns. The sign bias tests also return non-significant results, implying that the positive and negative shocks do not produce asymmetric volatility responses once the conditional variance is modeled. Taken together, these diagnostic tests indicate that the model fits the data well and successfully captures the core volatility dynamics.

The Nyblom stability test statistic (= 4.5232), however, suggests some evidence of parameter instability. The joint statistic substantially exceeds the corresponding critical values, and the individual statistics for $\hat{\alpha}_1$ and $\hat{\beta}_1$ (= 1.11146 and 0.65972 respectively) indicate structural shifts in volatility dynamics over the sample period. Although the Adjusted Pearson Goodness-of-Fit test rejects the null hypothesis of distributional adequacy under multiple grouping schemes, this does not necessarily invalidate the

model; rather, it highlights that even the Student-t distribution may not fully capture the extremity or asymmetry of Bitcoin's empirical tail behavior. Extensions such as the skewed-t distribution or more flexible volatility models may provide improved fit. We have not pursued that line of inquiry further in this work.

In summary, the estimated GARCH(1,1) model provides a statistically coherent representation of Bitcoin's return volatility. It effectively captures the high persistence and heavy-tailed nature of the series while producing well-behaved residuals and a satisfactory overall model fit.

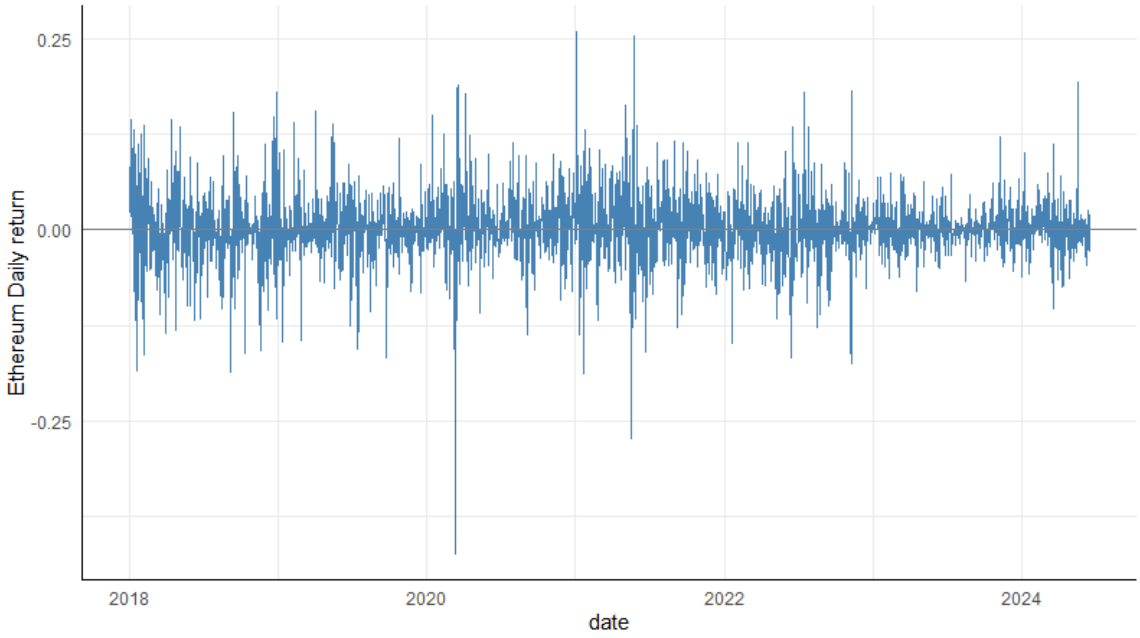


Figure 7.3: Time Series Plot of Ethereum Daily Returns

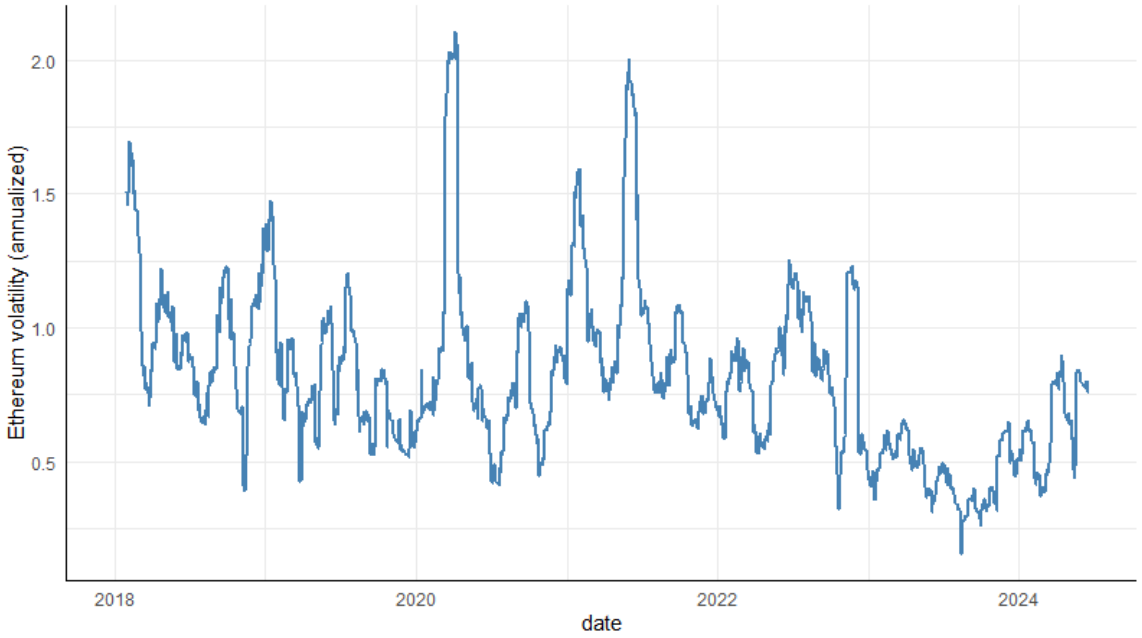


Figure 7.4: Monthly Volatility (Annualized) of Ethereum Returns

We subsequently analyze the daily returns of Ethereum, applying the same methodological framework employed for the Bitcoin return series. The time series plot in Figure 7.3 shows that Ethereum's daily returns fluctuate around zero with frequent sign changes and several spikes, particularly during major market disruptions in early 2020 and subsequent periods of market stress. This pattern indicates the evidence of volatility clustering, where large positive and negative returns occur in concentrated episodes. This behavior is further reflected in the 30-day rolling annualized volatility shown in Figure 7.4, which reveals substantial time variation in Ethereum's risk levels. As earlier in the case of Bitcoin, the volatility rises sharply during the periods of market turbulence, most notably in 2020, and remains elevated for sometime relative to other calmer periods where it more quickly declines toward lower levels.

Table 7.2: GARCH Model Fit for Ethereum Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	0.001073	0.000636	1.6860	0.091792
ω	0.000038	0.000019	1.9915	0.046428
α_1	0.102492	0.020880	4.9087	0.000001
β_1	0.896508	0.022157	40.4615	0.000000
shape	3.441106	0.254099	13.5424	0.000000
Information Criteria: Akaike = -3.6169, Bayes = -3.6047, Shibata = -3.6169 Hannan–Quinn = -3.6124, Log-Likelihood = 4272.934				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	0.0002234	0.988074		
Lag[$2(p+q)+(p+q)-1$][2]	2.8750	0.151700		
Lag[$4(p+q)+(p+q)-1$][5]	10.0200	0.009086		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	0.9713	0.500	2.000	0.3243
ARCH Lag[5]	1.8212	1.440	1.667	0.5121
ARCH Lag[7]	1.9123	2.315	1.543	0.7359
Nyblom Stability Test	Value			
Joint Statistic	2.2714			
Individual Statistics				
μ	0.1743			
ω	0.9741			
α_1	0.8261			
β_1	0.6960			
shape	0.7151			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	0.5321	0.5947		
Negative Sign Bias	0.0465	0.9629		
Positive Sign Bias	1.2713	0.2038		
Joint Effect	4.3523	0.2258		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	26.86	0.107851		
30	50.53	0.007923		
40	61.73	0.011664		
50	67.92	0.037960		

The GARCH estimation results for Ethereum returns, reported in Table 7.2, reveal a volatility process that is highly persistent and strongly driven by recent shocks. The estimate of mean parameter $\hat{\mu} = 0.001073$ is marginally significant at the 10% level, indicating that Ethereum's unconditional daily return is small and not reliably different from zero. The variance dynamics exhibits the expected GARCH structure, the estimate of ARCH coefficient $\hat{\alpha}_1 = 0.1025$ is positive and highly significant, showing that new return shocks exert an immediate impact on volatility, while the estimate of GARCH coefficient $\hat{\beta}_1 = 0.8965$ is also strongly significant and close to unity, implying a highly persistent volatility process with slow decay of shocks. The significant but small magnitude of $\hat{\omega}$ suggests a low unconditional volatility level, consistent with near-IGARCH behavior where recent shocks dominate long-run variance dynamics. The estimated shape parameter of the Student-t distribution, $\hat{\nu} \approx 3.44$, indicates extremely heavy tails, confirming the suitability of a heavy-tailed specification for Ethereum returns.

Model diagnostics indicate strong overall adequacy. The weighted Ljung-Box and ARCH LM tests produce large p-values across multiple lags, implying no remaining serial dependence or residual ARCH effects, and the sign-bias tests show no evidence of asymmetric volatility responses once conditional variance is modeled. However, the Nyblom stability test suggests potential parameter instability, with the joint statistic exceeding conventional critical values and several individual parameters approaching or surpassing their thresholds, indicating structural shifts in Ethereum's volatility dynamics over the sample. The Adjusted Pearson goodness-of-fit test shows mixed results, with rejection under more granular groupings, suggesting that although the Student-t distribution performs reasonably well, it may still fall short of fully capturing the extreme tail behavior of Ethereum. Overall, the fitted GARCH(1,1) model provides a

representation of Ethereum's volatility, capturing its heavy-tailed, shock-driven, and highly persistent nature.

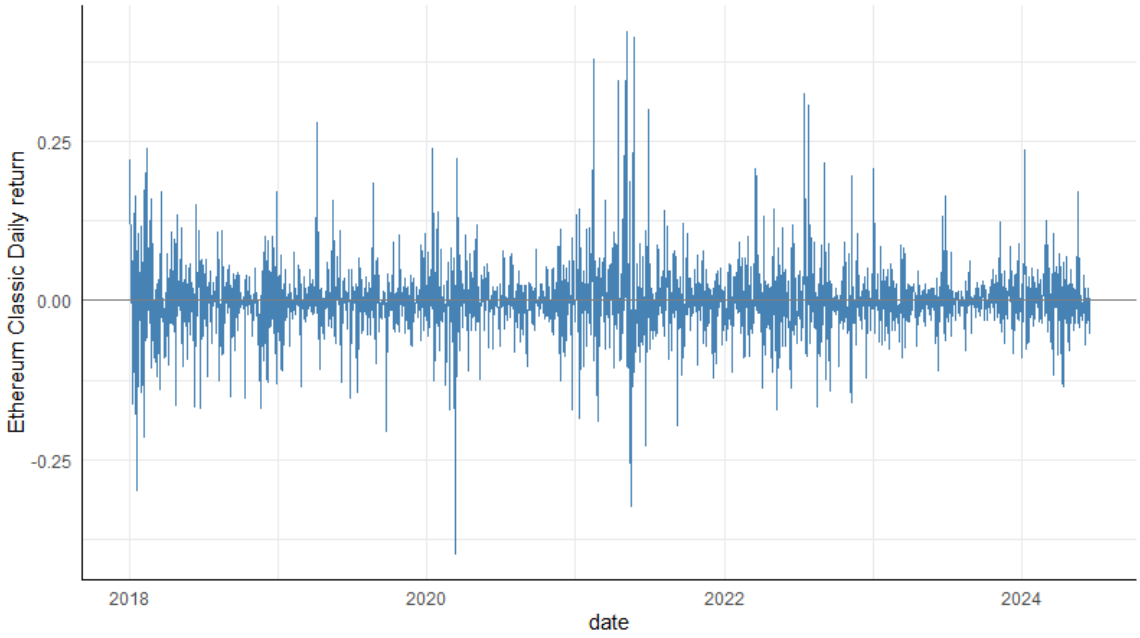


Figure 7.5: Time Series Plot of Ethereum Classic Daily Returns

For Ethereum Classic, the daily return plot in Figure 7.5 shows that the series oscillates around zero with frequent sign reversals and several pronounced spikes, particularly during periods of market turbulence in 2020 and 2021, indicating volatility clustering. The 30-day rolling annualized volatility shown in Figure 7.6, exhibits substantial time variation and sharp volatility surges, including episodes where annualized volatility exceeds 3.0. These patterns highlight Ethereum Classic's characteristically high and time-varying risk profile.

Table 7.3: GARCH Model Fit for Ethereum Classic Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	-0.000273	0.000685	-0.3985	0.690264
ω	0.000196	0.000065	3.0264	0.002475
α_1	0.209453	0.042977	4.8736	0.000001
β_1	0.789547	0.040945	19.2832	0.000000
shape	2.928044	0.177628	16.4841	0.000000
Information Criteria: Akaike = -3.3436, Bayes = -3.3314, Shibata = -3.3437 Hannan–Quinn = -3.3392, Log-Likelihood = 3950.504				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	0.09122	0.76263		
Lag[$2(p+q)+(p+q)-1$][2]	0.69763	0.60765		
Lag[$4(p+q)+(p+q)-1$][5]	7.07179	0.04982		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	1.499	0.500	2.000	0.2208
ARCH Lag[5]	1.880	1.440	1.667	0.4985
ARCH Lag[7]	4.205	2.315	1.543	0.3180
Nyblom Stability Test	Value			
Joint Statistic	2.0281			
Individual Statistics				
μ	0.05831			
ω	0.35016			
α_1	0.69827			
β_1	0.49244			
shape	0.33344			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	1.1900	0.2342		
Negative Sign Bias	0.9708	0.3317		
Positive Sign Bias	0.2601	0.7948		
Joint Effect	1.6433	0.6496		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	18.46	0.4921		
30	28.71	0.4801		
40	29.97	0.8502		
50	48.26	0.5029		

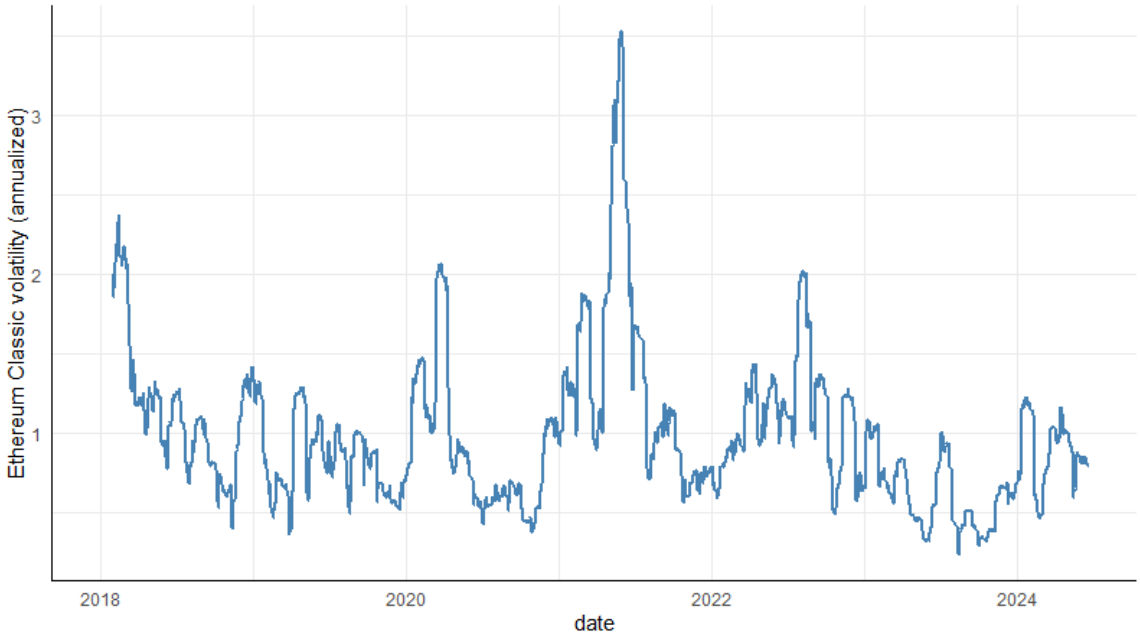


Figure 7.6: Monthly Volatility (Annualized) of Ethereum Classic Returns

The GARCH estimation results for Ethereum Classic are reported in Table 7.3. The mean parameter estimate $\hat{\mu} = -0.000273$ is not statistically significant, indicating that returns fluctuate around zero with no meaningful unconditional drift. The ARCH coefficient estimate $\hat{\alpha}_1 = 0.2095$ is large and highly significant, meaning that new shocks have an immediate and substantial impact on volatility, while the GARCH coefficient estimate $\hat{\beta}_1 = 0.7895$ is also significant and suggests considerable but comparatively faster decaying persistence than Bitcoin or Ethereum. The significant constant term estimate $\hat{\omega}$ indicates a relatively higher baseline level of unconditional variance. The estimated shapre parameter of the Student-t distribution, $\hat{\nu} \approx 2.93$, reflects extremely heavy tails, even lower than Bitcoin and Ethereum. Diagnostic tests show broadly satisfactory model performance. The weighted Ljung-Box and weighted ARCH LM tests detect no residual autocorrelation or remaining ARCH effects, and the sign-bias tests

indicate no asymmetric volatility responses. The Nyblom test yields a joint statistic above standard critical values, suggesting potential parameter instability over the sample, though most individual statistics remain below their thresholds. The Adjusted Pearson goodness-of-fit test does not reject the model under any grouping scheme, indicating that the Student-*t* distribution adequately captures the return distribution. Overall, the fitted GARCH(1,1) model provides a coherent and empirically consistent representation of Ethereum Classic's volatility dynamics.

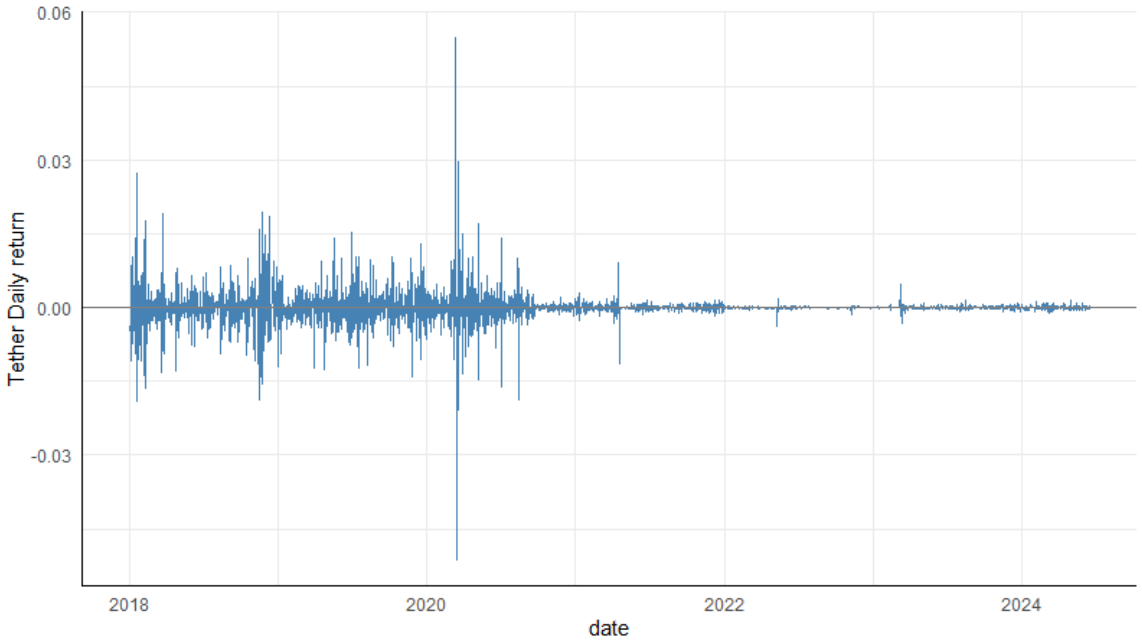


Figure 7.7: Time Series Plot of Tether Daily Returns

The Figures 7.7 and 7.8 show that Tether's daily returns remain tightly centered around zero, with only occasional deviations during periods of market stress in 2019 -

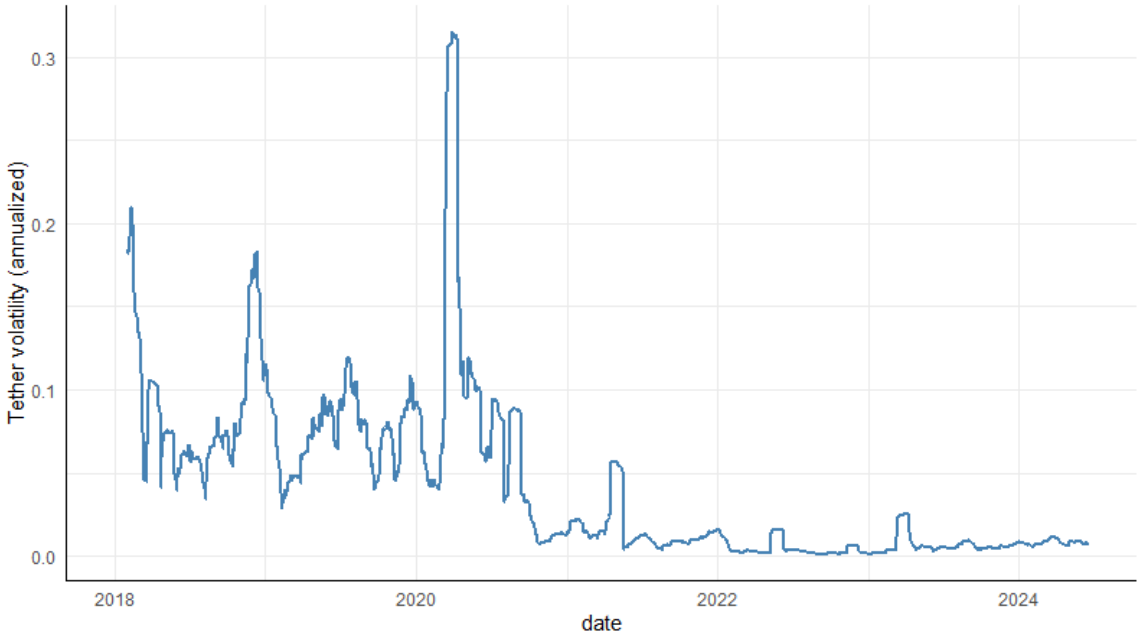


Figure 7.8: Monthly Volatility (Annualized) of Tether Returns

2020, followed by minimal fluctuations from 2021 onward. The 30-day rolling annualized volatility exhibits a similar pattern that brief but pronounced surges during stress episodes, followed by a rapid decline and a sustained period of exceptionally low volatility, reflecting the stabilization of Tether in more recent years.

The GARCH estimates for Tether is presented in Table 7.4. The estimate of mean is not significantly different from zero ($\hat{\mu} \approx 0$, $p = 0.979$) and the estimate of variance constant is effectively zero and insignificant ($\hat{\omega} \approx 0$, $p = 0.999$), whereas the volatility dynamics are statistically significant with $\hat{\alpha}_1 = 0.0660$ and $\hat{\beta}_1 = 0.8922$ (both $p < 0.001$), implying moderate shock impact with high persistence ($\hat{\alpha}_1 + \hat{\beta}_1 \approx 0.958$). The Student-$t$ degrees of freedom estimate $\hat{\nu} \approx 4.01$ indicate heavy tails.The weighted ARCH LM tests find no remaining ARCH effects (large p-values across lags), but weighted Ljung-Box tests reject no autocorrelation in the standardized residuals at multiple horizons ($p \approx 0$),

Table 7.4: GARCH Model Fit for Tether Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	0.000000	0.000004	-0.026887	0.97855
ω	0.000000	0.000000	0.001801	0.99856
α_1	0.066022	0.012168	5.426049	0.00000
β_1	0.892203	0.023817	37.460855	0.00000
shape	4.012213	0.175053	22.919983	0.00000
Information Criteria: Akaike = -10.990, Bayes = -10.978, Shibata = -10.990 Hannan–Quinn = -10.986, Log-Likelihood = 12973.32				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	67.36	2.22×10^{-16}		
Lag[$2(p+q)+(p+q)-1$][2]	69.72	0.00000		
Lag[$4(p+q)+(p+q)-1$][5]	72.67	0.00000		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	0.006304	0.500	2.000	0.9367
ARCH Lag[5]	0.043340	1.440	1.667	0.9960
ARCH Lag[7]	0.080721	2.315	1.543	0.9996
Nyblom Stability Test	Value			
Joint Statistic	384.6211			
Individual Statistics				
μ	0.01647			
ω	210.19545			
α_1	55.57887			
β_1	32.30047			
shape	9.96063			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	1.7577	0.07892		
Negative Sign Bias	0.7015	0.48306		
Positive Sign Bias	0.5512	0.58157		
Joint Effect	3.1091	0.37511		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	212.4	1.143×10^{-34}		
30	229.7	4.226×10^{-33}		
40	241.1	6.098×10^{-31}		
50	258.3	3.268×10^{-30}		

suggesting residual serial dependence in the mean not captured by a constant-mean specification. Parameter constancy is strongly violated that the Nyblom joint statistic far exceeds critical values. Distributional adequacy is also rejected overwhelmingly by the Adjusted Pearson test across all groupings ($p < 10^{-30}$), reflecting the minimal-variation nature of stablecoin returns that a standard Student-t cannot reproduce. Sign-bias tests are mostly insignificant, indicating little evidence of asymmetry once conditional variance is modeled. Overall, while the GARCH(1,1) captures the limited volatility clustering present, the diagnostics make clear that a stable asset like Tether is better modeled with frameworks allowing a near-constant variance.

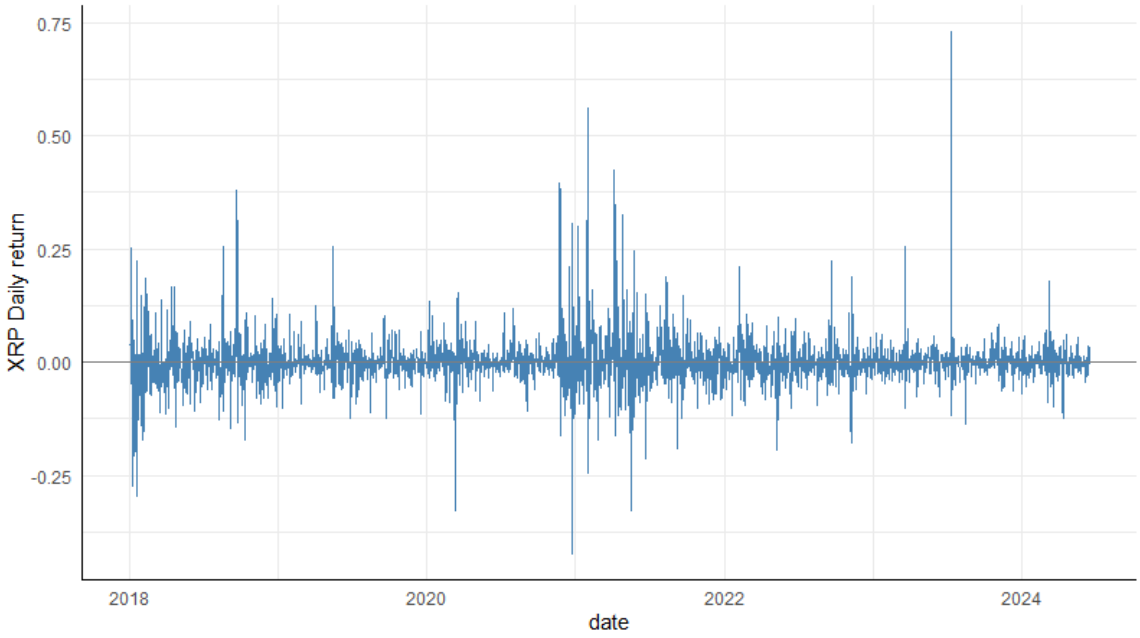


Figure 7.9: Time Series Plot of XRP Daily Returns

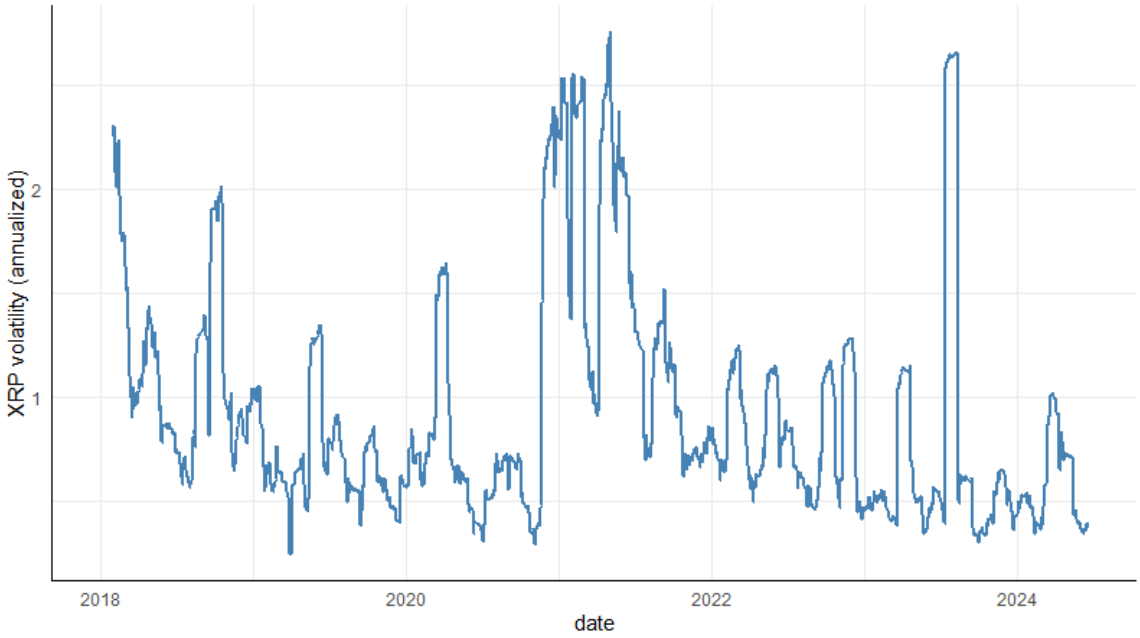


Figure 7.10: Monthly Volatility (Annualized) of XRP Returns

The Figures 7.9 and 7.10 indicate that XRP's daily returns are centered near zero with frequent sign reversals and several pronounced spikes, most notably in 2021 and later isolated episodes, providing clear evidence of volatility clustering. The 30-day rolling annualized volatility exhibits extended high-volatility regimes around 2021 with peaks above 2 on an annualized scale, and intermittent surges thereafter, underscoring pronounced time variation in risk.

The GARCH(1,1) results for XRP in Table 7.5 are consistent with these patterns. The estimate of mean parameter is insignificant ($\hat{\mu} = -0.000780$, $p = 0.182$), supporting a zero-mean specification for the conditional mean. The variance dynamics shows that the ARCH term estimate is large and highly significant ($\hat{\alpha}_1 = 0.2316$, $p < 10^{-6}$), indicating a strong immediate impact of new shocks on volatility, while the GARCH term estimate is also highly significant($\hat{\beta}_1 = 0.7674$, $p < 10^{-6}$). Their sum is approximately one

Table 7.5: GARCH Model Fit for XRP Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	-0.000780	0.000584	-1.3347	0.181962
ω	0.000170	0.000041	4.1371	0.000035
α_1	0.231630	0.041979	5.5177	0.000000
β_1	0.767370	0.032863	23.3503	0.000000
shape	2.889060	0.163308	17.6909	0.000000
Information Criteria: Akaike = -3.5994, Bayes = -3.5872, Shibata = -3.5994 Hannan–Quinn = -3.5949, Log-Likelihood = 4252.26				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	0.004822	0.9446		
Lag[$2(p+q)+(p+q)-1$][2]	0.328121	0.7803		
Lag[$4(p+q)+(p+q)-1$][5]	1.922146	0.6369		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	0.02188	0.500	2.000	0.8824
ARCH Lag[5]	0.05900	1.440	1.667	0.9938
ARCH Lag[7]	0.09592	2.315	1.543	0.9994
Nyblom Stability Test	Value			
Joint Statistic	2.5956			
Individual Statistics				
μ	0.1786			
ω	0.4159			
α_1	1.3782			
β_1	1.1878			
shape	0.6194			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	1.1142	0.2653		
Negative Sign Bias	0.7476	0.4548		
Positive Sign Bias	0.2759	0.7827		
Joint Effect	1.3147	0.7257		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	20.69	0.3539		
30	26.63	0.5918		
40	33.19	0.7316		
50	45.93	0.5983		

($\hat{\alpha}_1 + \hat{\beta}_1 \approx 0.999$), implying extremely persistent volatility with very slow decay of shocks. The estimate of variance constant is positive and significant ($\hat{\omega} = 0.000170$, $p = 3.5 \times 10^{-5}$), pointing to a non-negligible unconditional variance level. The Student-*t* degrees-of-freedom estimate ($\hat{\nu} \approx 2.89$) signals extremely heavy tails, consistent with the large return spikes observed in the plots. Model diagnostics are generally favorable. The weighted Ljung-Box tests yield large *p*-values across multiple lags indicating no remaining serial correlation, and the weighted ARCH LM tests do not detect residual ARCH effects. The sign-bias tests are insignificant implying no asymmetric volatility response once conditional variance is modeled. The Nyblom stability test rejects parameter constancy with joint statistic of 2.596 exceeding conventional critical values and individual statistics for α_1 and β_1 above the 1% threshold suggesting structural shifts in volatility dynamics over the sample. The Adjusted Pearson goodness-of-fit test does not reject distributional adequacy across all grouping schemes, indicating that the Student-*t* specification provides a reasonable fit to the return distribution. Overall, the fitted GARCH(1,1) model captures XRP's heavy tails, strong shock sensitivity, and extremely persistent volatility.

The Figures 7.11 and 7.12 indicate that Cardano's daily returns are centered near zero with frequent sign reversals and several pronounced spikes indicating volatility clustering. The 30-day rolling annualized volatility likewise shows extended high-volatility episodes in 2020 and 2021 followed by a moderation phase with recurrent but smaller surges thereafter, underscoring pronounced time variation in risk.

The GARCH estimates for Cardano is presented in Table 7.6. The mean estimate is insignificant ($\hat{\mu} = -0.000811$, $p = 0.293$), supporting a zero-mean conditional specification. The variance dynamics shows a highly significant ARCH term estimate ($\hat{\alpha}_1 = 0.164$, $p < 10^{-6}$), indicating a strong immediate effect of new shocks, and a

Table 7.6: GARCH Model Fit for Cardano Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	-0.000811	0.000771	-1.0524	0.292610
ω	0.000118	0.000039	3.0486	0.002299
α_1	0.163979	0.032483	5.0481	0.000000
β_1	0.826170	0.030934	26.7078	0.000000
shape	3.712035	0.302297	12.2794	0.000000
Information Criteria: Akaike = -3.2672, Bayes = -3.2550, Shibata = -3.2672 Hannan–Quinn = -3.2628, Log-Likelihood = 3860.318				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	0.006193	0.9373		
Lag[$2(p+q)+(p+q)-1$][2]	1.329703	0.4026		
Lag[$4(p+q)+(p+q)-1$][5]	5.350169	0.1274		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	0.9591	0.500	2.000	0.3274
ARCH Lag[5]	1.1785	1.440	1.667	0.6808
ARCH Lag[7]	1.6749	2.315	1.543	0.7857
Nyblom Stability Test	Value			
Joint Statistic	2.3219			
Individual Statistics				
μ	0.1486			
ω	1.6437			
α_1	0.5455			
β_1	1.1977			
shape	0.4954			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	0.6833	0.4945		
Negative Sign Bias	0.2880	0.7734		
Positive Sign Bias	0.3585	0.7200		
Joint Effect	1.1503	0.7649		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	17.25	0.57265		
30	38.02	0.12196		
40	41.73	0.35301		
50	66.31	0.05021		

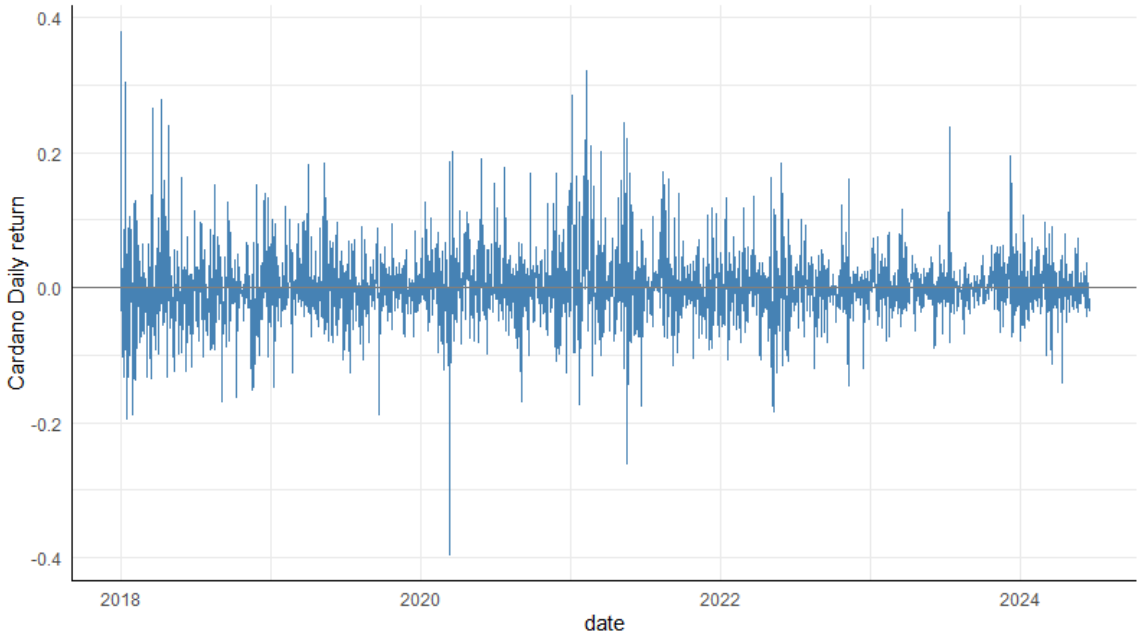


Figure 7.11: Time Series Plot of Cardano Daily Returns

strongly significant GARCH term estimate ($\hat{\beta}_1 = 0.826$, $p < 10^{-6}$). Their sum $\hat{\alpha}_1 + \hat{\beta}_1 \approx 0.990$ implies very persistent volatility with slow decay of shocks, though somewhat less extreme than assets whose sums are effectively unity. The estimate of variance constant is positive and significant ($\hat{\omega} = 0.000118$, $p = 0.0023$), pointing to a nontrivial long-run variance level. The Student-t degrees of freedom estimate ($\hat{\nu} \approx 3.71$) indicate heavy tails, consistent with the return spikes seen in the plots. Model diagnostics shows that the weighted Ljung-Box tests yield nonsignificant results across multiple lags, and weighted ARCH LM tests detect no remaining ARCH effects. The sign-bias tests are also insignificant, suggesting no asymmetric volatility response once conditional variance is modeled. The Nyblom stability test rejects parameter constancy indicating structural shifts in the volatility process over the sample. Distributional adequacy is generally

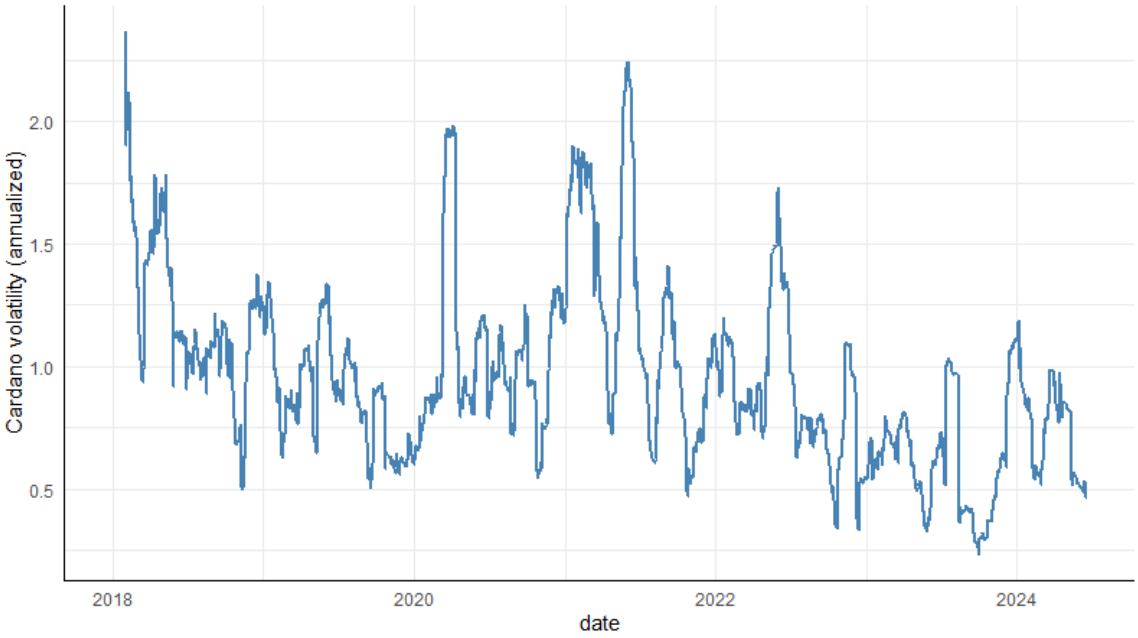


Figure 7.12: Monthly Volatility (Annualized) of Cardano Returns

acceptable as shows in the Adjusted Pearson test. Overall, the fitted GARCH(1,1) model captures Cardano's heavy tails, strong shock sensitivity, and high volatility persistence.

The Figures 7.13 and 7.14 highlight the distinctive return behavior of Dogecoin. The daily return series fluctuates around zero with frequent sign reversals, but unlike the other major cryptocurrencies, it features several exceptionally large spikes during the speculative surge of early 2021, indicating extreme episodes of market exuberance and clear volatility clustering. The 30-day rolling annualized volatility plot exhibits dramatic surges, including an unprecedented spike exceeding 12 on an annualized scale, followed by a rapid collapse and sporadic smaller bursts thereafter.

The GARCH(1,1) estimates for Dogecoin reported in Table 7.7 are consistent with these empirical features. The estimate of mean return is small but statistically significant ($\hat{\mu} = -0.001237$, $p = 0.032$), though this likely reflects sampling variation rather than a

Table 7.7: GARCH Model Fit for Dogecoin Returns

Parameter	**Estimate**	**Std. Error**	**t value**	**P value**
μ	-0.001237	0.000577	-2.1427	0.032138
ω	0.000224	0.000053	4.1940	0.000027
α_1	0.339706	0.045332	7.4937	0.000000
β_1	0.659294	0.040941	16.1034	0.000000
shape	3.053905	0.153719	19.8668	0.000000
Information Criteria: Akaike = -3.4673, Bayes = -3.4551, Shibata = -3.4673 Hannan–Quinn = -3.4628, Log-Likelihood = 4096.385				
Weighted Ljung–Box Test	Statistic	P value		
Lag[1]	0.09179	0.7619		
Lag[$2(p+q)+(p+q)-1$][2]	0.74006	0.5908		
Lag[$4(p+q)+(p+q)-1$][5]	1.38622	0.7678		
Weighted ARCH LM Tests	Statistic	Shape	Scale	P value
ARCH Lag[3]	0.002088	0.500	2.000	0.9636
ARCH Lag[5]	0.006643	1.440	1.667	0.9997
ARCH Lag[7]	0.010108	2.315	1.543	1.0000
Nyblom Stability Test	Value			
Joint Statistic	6.3031			
Individual Statistics				
μ	0.2320			
ω	0.4658			
α_1	1.2683			
β_1	0.5421			
shape	0.5915			
Asymptotic Critical Values (10%, 5%, 1%)				
Joint:	1.28	1.47	1.88	
Individual:	0.35	0.47	0.75	
Sign Bias Test	t value	Prob		
Sign Bias	0.18736	0.85139		
Negative Sign Bias	2.01837	0.04367		
Positive Sign Bias	0.01121	0.99106		
Joint Effect	5.10153	0.16451		
Adjusted Pearson Goodness-of-Fit Test				
Group	Statistic	P value		
20	35.07	0.01371		
30	36.24	0.16675		
40	50.00	0.11152		
50	52.42	0.34299		

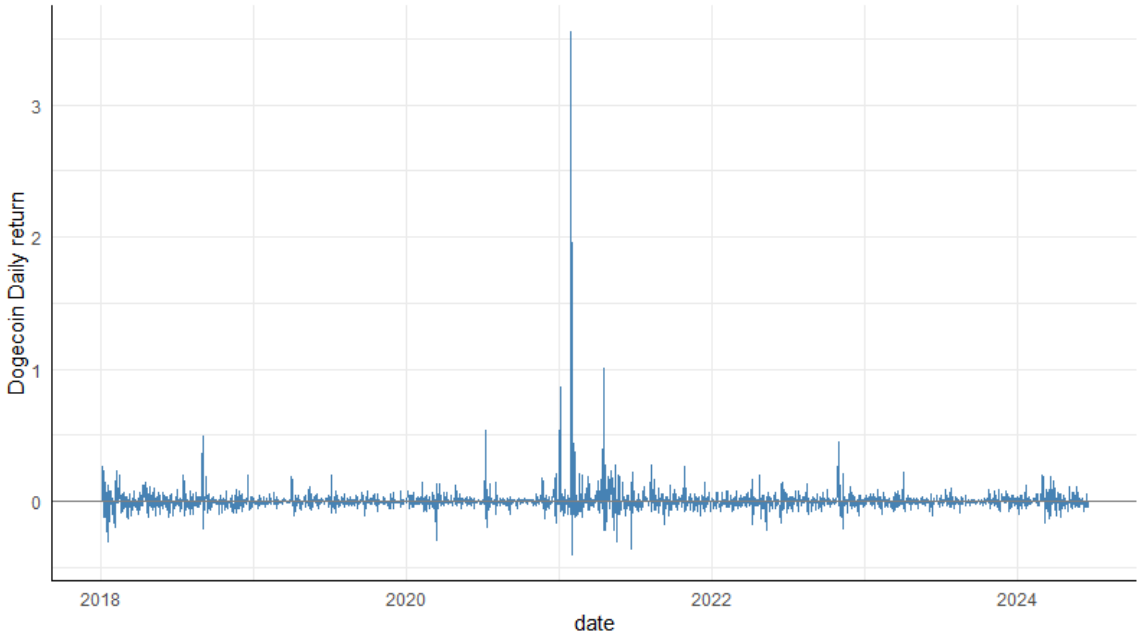


Figure 7.13: Time Series Plot of Dogecoin Daily Returns

persistent drift. The variance dynamics show a strong GARCH structure. The estimate of ARCH coefficient is large and highly significant ($\hat{\alpha}_1 = 0.3397$, $p < 10^{-6}$), implying that new shocks have an immediate and substantial impact on volatility, while the GARCH term estimate remains strongly significant ($\hat{\beta}_1 = 0.6593$, $p < 10^{-6}$). The sum of the coefficients ($\hat{\alpha}_1 + \hat{\beta}_1 \approx 0.999$) suggests that volatility is extremely persistent, with shocks decaying only very slowly. The degrees-of-freedom parameter estimate for the Student-t distribution ($\hat{\nu} \approx 3.05$) confirms the presence of very heavy tails, consistent with the large jumps observed in the time-series plots. Diagnostic tests show that the weighted Ljung-Box tests and weighted ARCH LM tests yield large p-values across multiple lags, indicating no remaining autocorrelation or residual ARCH effects. However, the Nyblom stability test rejects parameter constancy. The sign-bias tests show one marginally significant case (negative sign bias, $p = 0.044$), hinting at a potentially asymmetric

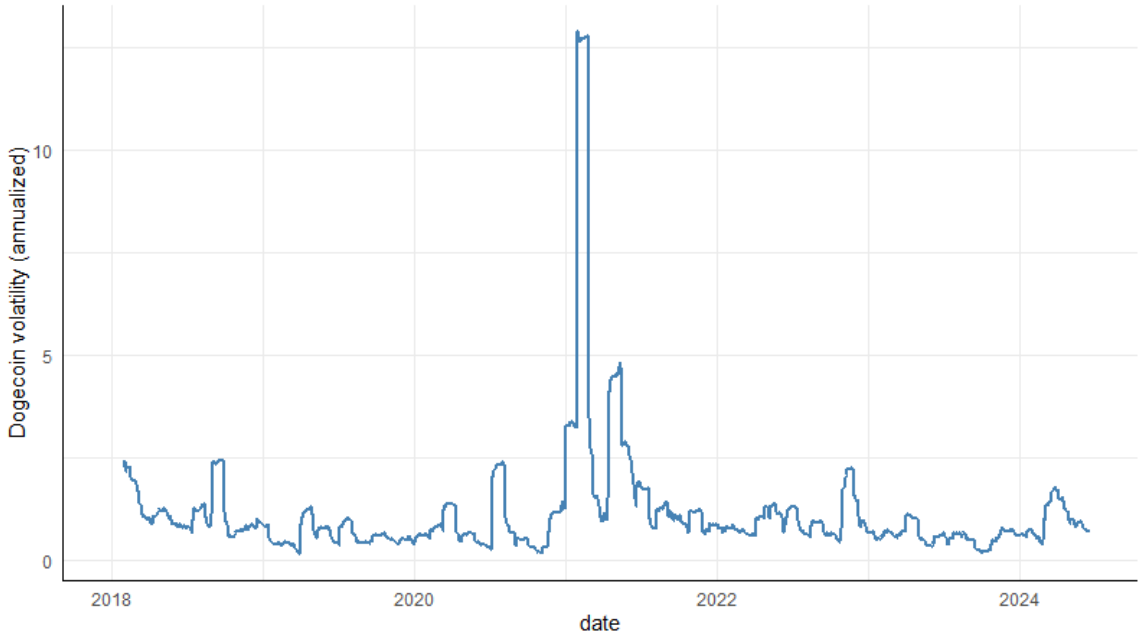


Figure 7.14: Monthly Volatility (Annualized) of Dogecoin Returns

response to negative shocks, although the joint test remains insignificant. The Adjusted Pearson goodness-of-fit test yields mixed results, rejecting at the 20-group level but not at higher groupings, indicating that while the Student-*t* distribution provides a reasonable approximation, it may not fully capture the extreme tail behavior of Dogecoin. Overall, the GARCH(1,1) model captures the Dogecoin's heavy-tailed, shock-sensitive, and highly persistent volatility structure.

7.3 Multivariate GARCH Model

The multivariate GARCH framework is employed when analyzing joint risk or interdependence across multiple financial time series. It models not only the individual volatilities of each series but also their time-varying covariance structure. The conditional covariance matrix is defined as

$$H_t = \text{var}(\varepsilon_t), \qquad \varepsilon_t \in \mathbb{R}^N$$

Among multivariate GARCH specifications, the BEKK model, named after Baba, Engle, Kraft, and Kroner, is widely used for capturing the dynamics of time-varying covariance matrices across multiple returns, see details in Engle and Kroner (1995). It jointly models multivariate volatility, allows correlations to evolve over time, and permits volatility spillovers across assets. For example, a shock to asset A can increase the conditional variance of asset B, reflecting interconnected market behavior.

Suppose $r_t \in \mathbb{R}^n$ denotes an n-dimensional vector of asset returns. The multivariate mean equation is specified as

$$r_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim N(0, H_t) \tag{7.5}$$

The conditional covariance matrix H_t follows the BEKK(p,q,K) specification, defined as

$$H_t = CC' + \sum_{k=1}^{K}\sum_{i=1}^{q} A'_{ik}\,\varepsilon_{t-i}\varepsilon'_{t-i}A_{ik} + \sum_{k=1}^{K}\sum_{i=1}^{p} G'_{ik}\,H_{t-i}\,G_{ik}, \tag{7.6}$$

where C, A_{ik}, and G_{ik} are $N \times N$ parameter matrices, with C constrained to be lower triangular to ensure that H_t is positive definite for all t.

The most commonly used specification is the BEKK$(1,1,1)$ model, which relies only on one-period lags, defined as

$$H_t = C'C + A'\varepsilon_{t-1}\varepsilon'_{t-1}A + G'H_{t-1}G \tag{7.7}$$

where

- H_t is the conditional covariance matrix at time t.
- ε_{t-1} denotes the vector of lagged shocks.
- C is a lower triangular matrix ensuring positive definiteness of H_t.
- A represents the ARCH effects, capturing how past shocks affect current covariance.
- G represents the GARCH effects, capturing the persistence of past covariances.

In our analysis, we estimate the symmetric BEKK model using the R package *BEKKs*, see details in Fülle et al. (2024). The model is applied to daily returns of three major cryptocurrencies: Bitcoin, Ethereum, and Ethereum Classic.

As presented in Table 7.8, the symmetric BEKK(1,1) model fits the joint dynamics of Bitcoin, Ethereum, and Ethereum Classic well. The process is stationary, and the optimizer converges quickly with 19 BHHH iterations, while the information criteria are favorable (AIC $= -29904.11$, BIC $= -29890.51$). The matrix C has most elements statistically significant, indicating a well-defined unconditional covariance structure.

In the shock effects matrix A, significant diagonal elements confirm strong own-shock effects, while several significant off-diagonal elements (such as $\hat{A}_{12}$, $\hat{A}_{13}$, and $\hat{A}_{21}$) indicate meaningful cross-market spillovers, suggesting that shocks in Bitcoin and Ethereum propagate to the other assets.

The persistence matrix G exhibits diagonal terms close to unity and multiple significant off-diagonal entries (such as $\hat{G}_{12}$, $\hat{G}_{21}$, $\hat{G}_{31}$, and $\hat{G}_{32}$). This pattern implies that both variances and covariances are highly persistent, and that periods of elevated co-movement tend to endure, reflecting strong temporal dependence in the multivariate volatility process.

Table 7.8: BEKK Model Estimation Results for Bitcoin, Ethereum and Ethereum Classic

Model Summary			
Log-Likelihood	14973.05	BEKK Model Stationary	TRUE
AIC	-29904.11	BHHH Iterations	19
BIC	-29890.51		
Parameter Matrix C	**1**	**2**	**3**
1	0.00579	0.00356	0.01062
2	0.00000	0.00407	0.01114
3	0.00000	0.00000	0.00008
t-values for C	**1**	**2**	**3**
1	12.45559	4.68918	9.31246
2	0.00000	11.03597	9.26879
3	0.00000	0.00000	0.00044
Parameter Matrix A	**1**	**2**	**3**
1	0.27117	-0.14992	-0.16286
2	-0.07857	0.32097	0.00130
3	0.01926	-0.01552	0.38173
t-values for A	**1**	**2**	**3**
1	13.17870	6.44009	5.01256
2	4.30359	14.75162	0.04324
3	1.82760	1.23936	24.67490
Parameter Matrix G	**1**	**2**	**3**
1	0.91992	0.02133	0.00759
2	0.02715	0.93977	0.01538
3	0.01403	0.01754	0.89572
t-values for G	**1**	**2**	**3**
1	100.81165	2.06692	0.52991
2	3.81024	117.68888	1.41366
3	3.05827	3.11025	139.78937

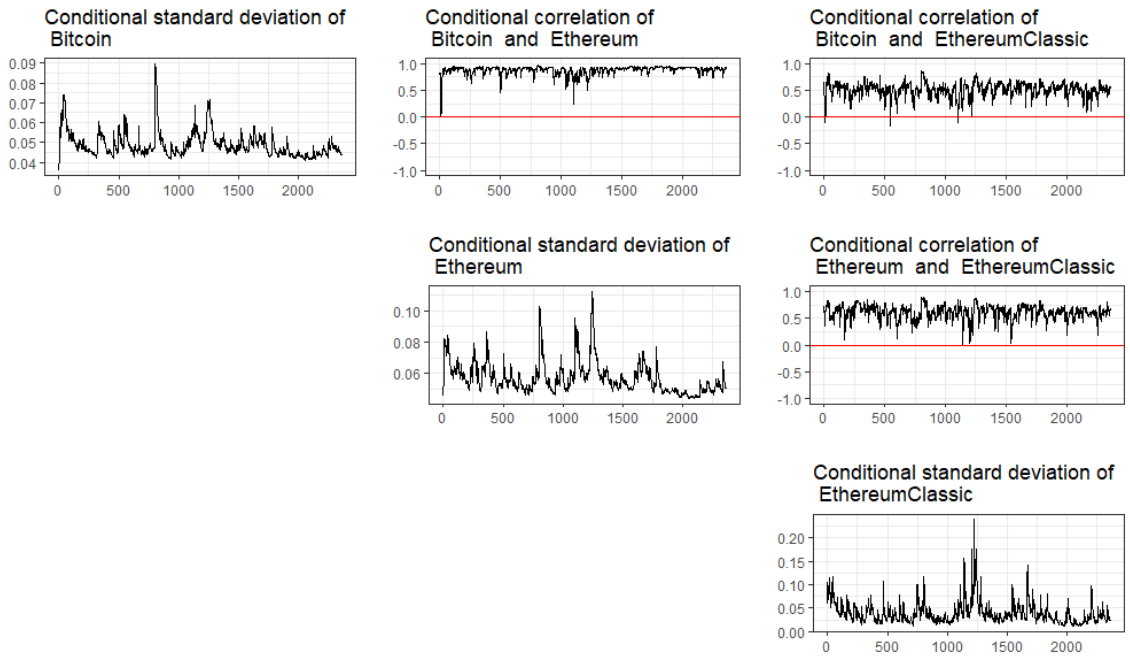


Figure 7.15: BEKK Model Result for Bitcoin, Ethereum and Ethereum Classic

The conditional standard deviations for all three assets exhibit pronounced clustering with slow mean reversion, as presented in Figure 7.15. Bitcoin and Ethereum show broadly similar volatility regimes with multiple cycles of surge and decay, while Ethereum Classic shows the largest relative spikes, reflecting its smaller market depth and greater sensitivity to market dislocations. Conditional correlations are persistently positive and time varying, particularly high between Bitcoin and Ethereum. Correlations involving Ethereum classic are also positive and time-varying, but comparatively more volatile. In practical terms, portfolio risk in this three coin set is dominated by shared factors and persistent volatility and co volatility regimes: shocks in the leaders (notably Bitcoin) propagate quickly across the complex and then decay only gradually.

7.4 Concluding Remarks

This chapter has examined the time-varying volatility dynamics of major cryptocurrencies using both univariate and multivariate GARCH models. Across all assets, the analysis highlights several main features of cryptocurrency returns, including pronounced volatility clustering, strong persistence of shocks, and heavy-tailed return distributions.

The univariate GARCH analyses reveal that the key cryptocurrencies exhibit highly persistent volatility, with $\alpha + \beta$ values close to unity. This indicates that volatility shocks decay slowly and that periods of elevated uncertainty tend to persist for extended durations. Assets such as Bitcoin and Ethereum display near-IGARCH behavior, while smaller assets like Ethereum Classic, Cardano, and Dogecoin exhibit even larger immediate shock responses, reflecting their higher sensitivity to market turbulence. In every case, the estimated parameter from the Student-*t* distribution confirm the presence of highly heavy tails, underscoring the prevalence of large and abrupt market moves. Diagnostic tests generally support the adequacy of the GARCH(1,1) framework for capturing the volatility structure, though some assets show signs of parameter instability.

The multivariate BEKK model further demonstrates strong interdependence across major cryptocurrencies. Significant off-diagonal elements in both the shock matrix A and the persistence matrix G provide clear evidence of volatility spillovers and time-varying co-movement. Shocks originating in Bitcoin and Ethereum propagate rapidly to Ethereum Classic, and the conditional correlations remain positive and strongly persistent throughout the sample. These results highlight the interconnected nature of cryptocurrency markets, where systemic shocks tend to transmit quickly across assets.

Overall, the findings in this chapter demonstrate that GARCH-type models offer a coherent and effective framework for capturing the volatility characteristics of

cryptocurrencies. The combination of high persistence, heavy tails, and significant cross-market spillovers underscores the need for robust, time-varying volatility models in risk assessment, derivative pricing, and portfolio optimization.

CHAPTER EIGHT

PREDICTION OF INTERRUPTIONS IN ENERGY SUPPLY

8.1 Introduction

It is practically impossible to imagine the modern civilizations and societies without energy. Energy is what runs the industry, helps the agriculture, lightens our world and runs the appliances. Energy is what charges our computers and cell phones and runs our automobiles. Energy is what helps our hospitals run smoothly where patient on inevitably dependent on medical equipment which run on electricity. Any interruption to energy supply puts our lives into tailspin and if this interruption is for an extended period then losses to business can amount to hundreds of millions of dollars. A case in point is the big power outage of Aug 14, 2003 in the USA when a transmission fault in Ohio was caused by contact with a tree. The domino effect was what became one of the largest outages ever in North America, affecting more than fifty million people in eight states in USA and the Ontario province of Canada. Economic losses were estimated to be at least 10 billions, according to the Electricity Consumers Resource Council . Similarly, a winter storm in Texas in February 2021 caused a wide spread black out when the state's electric grid operator ceased to adequately supply power in the extreme cold weather. As a result millions were left without heat for days in the frigid temperatures.

These events and other similar episodes all over the world underscore the importance of the problem of prediction of interruptions in the power supply. Needless to say that any such prediction is helpful in advance planning and in taking the precautionary actions to minimize the losses - economic, of lives or otherwise.

This article deals with one such problem for the state of Michigan which has a very unstable weather, with rains and storms all through the year and heavy snow fall and

extremely cold weather during winters. The DTE Energy is one of the main energy suppliers in the state and routinely faces power interruptions throughout the year. Needless to say, that prior signals for such interruptions are weak and further these signals are many. How do we then arrive at a reasonable prediction? Furthermore, can there be a machine learning approach where one can learn from a model which suitably assembles the information in various variables, which are unequal in quality?

To address the above question and to forecast daily interruptions in the supply of energy and identify the significant variables that influence outage volume using various weather variables and reliability program data, this article focuses on the application of various post-shrinkage strategies. We apply these strategies to select the optimal set of predictors and estimate the corresponding coefficients for accurately predicting the number of interruptions and provide insights into the effectiveness of programs in improving the service to the energy customers.

8.2 Description of the Problem and Variables

The problem here is to predict the daily interruptions to the power supply as defined by the daily count of sustained power interruptions in the whole electricity service territory of the company. We will denote it by $\mathscr{Y}$. To better forecast the daily interruptions, we take into account various factors that may influence the outcome. These factors include a range of weather variables, and reliability upgrade program variables. In addition, to account for other aspects such as seasonality, we also define variables for week and month. The following 73 predictors were considered and will be referred to as $X_1, X_2, ..., X_{73}$ throughout this article. These are listed in Table 8.1.

Table 8.1: Variable Definitions

X_1	= Maximum Air Temperature [F]

X_2 = Minimum Air Temperature [F]

X_3 = Maximum Dew Point [F]

X_4 = Minimum Dew Point [F]

X_5 = Daily Precipitation [inch]

X_6 = Average Wind Speed [knots]

X_7 = Average Wind Direction [deg]

X_8 = Minimum Relative Humidity [percentage]

X_9 = Average Relative Humidity [percentage]: computed by time averaging the available reports, it is not average of the daily max/min

X_{10} = Maximum Relative Humidity [percentage]

X_{11} = Reported Snowfall [inch]

X_{12} = Reported Snow Depth [inch]

X_{13} = Maximum sustained wind speed in knots

X_{14} = Maximum wind gust in knots

X_{15} = 1 when lightning occurs, 0 otherwise

X_{16} = 1 when freezing rain occurs, 0 otherwise

X_{17} = 1 when moderate rain occurs, 0 otherwise

X_{18} = 1 when heavy rain occurs, 0 otherwise

X_{19} = 1 when light rain occurs, 0 otherwise

X_{20} = 1 when moderate thunderstorm occurs, 0 otherwise

X_{21} = 1 when heavy thunderstorm occurs, 0 otherwise

X_{22} = 1 when light thunderstorm occurs, 0 otherwise

X_{23} = 1 when moderate thunderstorm with rainfall occurs, 0 otherwise

X_{24} = 1 when light thunderstorm with rainfall occurs, 0 otherwise

X_{25} = 1 when freezing drizzle occurs, 0 otherwise

X_{26} = 1 when moderate drizzle occurs, 0 otherwise

X_{27} = 1 when heavy drizzle occurs, 0 otherwise

X_{28} = 1 when light drizzle occurs, 0 otherwise

X_{29} = 1 when squalls occurs, 0 otherwise

X_{30} = 1 when heavy hail occurs, 0 otherwise

X_{31} = 1 when light hail occurs, 0 otherwise

X_{32} = 1 when Cloud-to-cloud lightning occurs, 0 otherwise

X_{33} = 1 when Cloud-to-air lightning occurs, 0 otherwise

X_{34} = 1 when Cloud-to-ground lightning occurs, 0 otherwise

X_{35} = 1 when ice pellets occur, 0 otherwise

X_{36} = 1 when snow pellets occur, 0 otherwise

$X_{37} - X_{41}$ = Number of reliability upgrade program A performed in the last 3 months, 6 months, 1,2 and 3 years

$X_{42} - X_{46}$ = Number of reliability upgrade program B performed in the last 3 months, 6 months, 1,2 and 3 years

$X_{47} - X_{51}$ = Number of reliability upgrade program C performed in the last 3 months, 6 months, 1,2 and 3 years

$X_{52} - X_{56}$ = Number of reliability upgrade program D performed in the last 3 months, 6 months, 1,2 and 3 years

$X_{57} - X_{61}$ = Number of reliability upgrade program E performed in the last 3 months, 6 months, 1,2 and 3 years

$X_{62} - X_{66}$ = Number of reliability upgrade program F performed in the last 3 months, 6 months, 1 ,2 and 3 years

X_{67} = 1 when small storm occurs, 0 otherwise

X_{68} = 1 when median storm occurs, 0 otherwise

X_{69} = 1 when large storm occurs, 0 otherwise

X_{70} = 1 when catastrophic level 1 storm occurs, 0 otherwise

X_{71} = 1 when catastrophic level 2 storm occurs, 0 otherwise

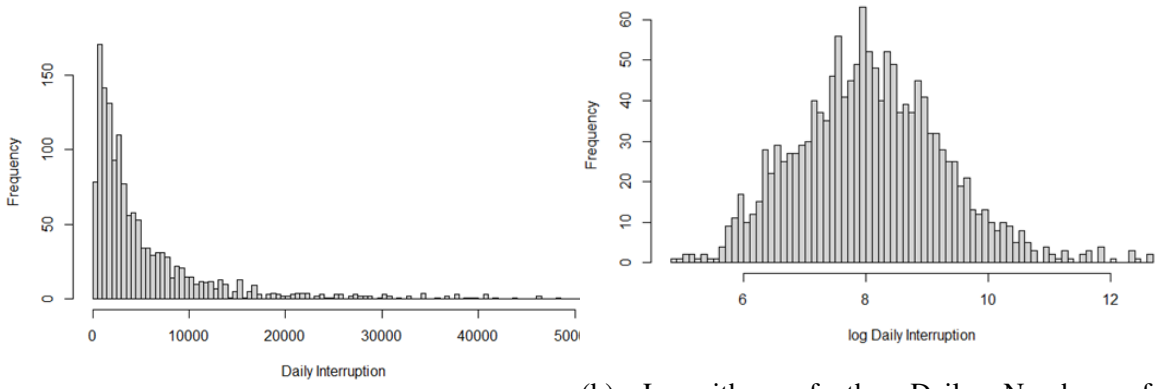
X_{72} = Periodic value for week-number of the year, calculated as week of year $\times\frac{2\pi}{53}$ for the year 2020 and 2021, while week-number of year $\times\frac{2\pi}{52}$ for other years

X_{73} = Periodic value for month, calculated as month $\times\frac{2\pi}{12}$.

The dataset utilized in this analysis is sourced from the DTE Electric Company and comprises outage data for a period of year 2020 to 2023. For the response Variable $\mathscr{Y}$, number of daily interruptions as reported by the smart meter for the whole service territory, a histogram is shown in Figure 8.1a. Clearly the distribution of the response variable is highly skewed. Taking logarithm of $\mathscr{Y}$, somewhat improves the skewness. The corresponding distribution shown in Figure 8.1b looks more symmetric. Nonetheless, asymmetry still cannot be completely overlooked. The Shapiro-Wilks Normality test confirms that, but we could see from the Figure 8.2 that it is close to the normal distribution after excluding a few outliers. That is however not a major problem since we intent to predict interruptions using some or all of the predictors and hence we are more concerned about the conditional distribution of the natural logarithm of number of daily interruptions, namely $log(\mathscr{Y})$. Let us denote it by $\mathbf{Y}$.

8.3 The Analysis of the Data

We will utilize post shrinkage strategies and penalized methods to fit the data and then compare the average prediction errors of the resulting prediction equations to identify the optimal choices. To achieve this, we will employ the criteria of AIC, BIC and LASSO and accordingly construct the shrinkage and positive shrinkage estimators using



(a) Daily Number of Interruptions.

(b) Logarithm of the Daily Number of Interruptions.

Figure 8.1: Histograms of Daily Number of Interruptions and Logarithm of the Daily Number of Interruptions

submodels obtained from these techniques. Equipped with all of the above machinery we now describe our analyses. Specifically, we first start with the full model.

8.3.1 Full Model Estimation

We first fit the full model that incorporates all independent variables. The coefficients obtained from this model will be utilized to construct the shrinkage estimators. Table 8.2 provides the estimates of the coefficients for the full model along with various other statistics.

Table 8.2: Estimates of the Regression Coefficients for the Full Model

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
(Intercept)	-1.4349058	2.3623973	0.543689	$R^2 = 0.5891$
X_1	0.0033218	0.0071929	0.64429	$Adjusted\ R^2 = 0.5675$
X_2	0.0152727	0.0074607	0.040837	$F = 27.28$

Continued on next page

Table 8.2 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
X_3	0.0171682	0.0075468	0.023064	$RootSqErr = 0.7954$
X_4	-0.0126686	0.0077311	0.101511	$pvalue < 2.2e^{-16}$
X_5	0.3988488	0.10746	0.000214	
X_6	0.0279179	0.0133763	0.03706	
X_7	0.0001618	0.0002457	0.5103	
X_8	-0.0059198	0.0049535	0.232259	
X_9	0.0132085	0.0082315	0.108803	
X_{10}	0.0040757	0.0052238	0.435396	
X_{11}	0.0088928	0.0416701	0.831039	
X_{12}	0.0364021	0.0194084	0.060922	
X_{13}	0.0264371	0.0112938	0.01938	
X_{14}	0.0079774	0.0060721	0.189141	
X_{15}	0.2636743	0.0961351	0.006171	
X_{16}	0.1810161	0.3024125	0.549555	
X_{17}	0.4021686	0.261641	0.124496	
X_{18}	0.1451987	0.1026169	0.157305	
X_{19}	-0.3293907	0.2613644	0.207782	
X_{20}	-0.0291328	0.2104974	0.889944	
X_{21}	0.1767641	0.1659967	0.287122	
X_{22}	1.5508562	0.8865858	0.080469	
X_{23}	-0.1870241	0.2884277	0.516817	
X_{24}	-1.5606427	0.9168795	0.088956	

Continued on next page

Table 8.2 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
X_{25}	0.095028	0.2472978	0.700841	
X_{26}	-0.0510137	0.1684587	0.762068	
X_{27}	0.0379124	0.5855984	0.948389	
X_{28}	0.1179744	0.1631385	0.469707	
X_{29}	0.2636193	0.2636098	0.317467	
X_{30}	-0.0352745	0.3762723	0.925323	
X_{31}	-0.6766859	0.6109972	0.268265	
X_{32}	-0.0611754	0.1361935	0.653372	
X_{33}	-0.2504114	0.2201783	0.255604	
X_{34}	0.1893963	0.1870241	0.311387	
X_{35}	-0.2476882	0.2128866	0.244837	
X_{36}	1.4986204	0.8483475	0.077529	
X_{37}	0.0337606	0.0134764	0.012353	
X_{38}	0.0323707	0.0192738	0.093276	
X_{39}	0.0324381	0.021364	0.129153	
X_{40}	0.0191274	0.0218948	0.382486	
X_{41}	-0.0088402	0.0243781	0.716936	
X_{42}	0.0002045	0.0018174	0.910417	
X_{43}	-0.0008144	0.0020722	0.694384	
X_{44}	0.0003854	0.0019128	0.840355	
X_{45}	-0.0002506	0.0010334	0.808463	
X_{46}	-0.0009193	0.0007443	0.216969	

Continued on next page

Table 8.2 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
X_{47}	0.0071749	0.0036609	0.050209	
X_{48}	0.0025333	0.0060921	0.677597	
X_{49}	0.0125848	0.0069474	0.070288	
X_{50}	0.0186308	0.0067421	0.005797	
X_{51}	0.0102931	0.0045987	0.025361	
X_{52}	0.0050599	0.0033574	0.132019	
X_{53}	-0.0039514	0.0034688	0.254848	
X_{54}	-0.0006412	0.0038367	0.867296	
X_{55}	0.0019115	0.0038731	0.621724	
X_{56}	0.0040732	0.0033033	0.217762	
X_{57}	-0.0316439	0.0178022	0.075701	
X_{58}	-0.0038722	0.0214933	0.857054	
X_{59}	-0.0262507	0.0230296	0.254538	
X_{60}	-0.0360484	0.0212406	0.089894	
X_{61}	-0.0562926	0.0230569	0.014752	
X_{62}	-0.0006363	0.0363784	0.986047	
X_{63}	-0.1390423	0.0620756	0.025256	
X_{64}	-0.2455342	0.092945	0.008341	
X_{65}	0.0534835	0.0909456	0.556573	
X_{66}	0.0383318	0.0733282	0.601237	
X_{67}	0.6252392	0.1516219	0.0000395	
X_{68}	0.9121814	0.1487507	1.13×10^{-9}	

Continued on next page

Table 8.2 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
X_{69}	0.7586724	0.2111241	0.000338	
X_{70}	1.0505726	0.1215712	$< 2 \times 10^{-16}$	
X_{71}	1.5199993	0.1413784	$< 2 \times 10^{-16}$	
X_{72}	-0.0024122	0.2694237	0.992858	
X_{73}	-0.0619809	0.1940962	0.749524	

Upon examining the output, we identify several statistically significant variables that considerably impact the response of interest, for example X_3, X_5, X_{13}. Additionally, there are variables that exhibit a moderate influence on the response, such as X_{12}, X_{47}, and X_{49}. However, certain variables, such as X_1, X_{11}, X_{73}, display a weak influence on the response variable.

The Figure 8.3a provides the residual plot, we see that the points are randomly scattered around the horizontal line. This indicates that the model successfully captures the underlying relationships in the dataset. The Q-Q plot in Figure 8.3b compares empirical quantiles of the residuals to the theoretical quantiles of a normal distribution. The points generally fall along the diagonal line. This suggests that the residuals are approximately normally distributed.

The Figure 8.3c shows the square root of the absolute standardized residuals and the fitted values to assess the homoscedasticity assumption of a linear regression model. There is no trend in the plot thereby indicating the homoscedasticity. This suggests that the variance of the residuals remains relatively constant. The Figure 8.3d helps identify influential observations that may have a large impact on the model's fit and conclusions.

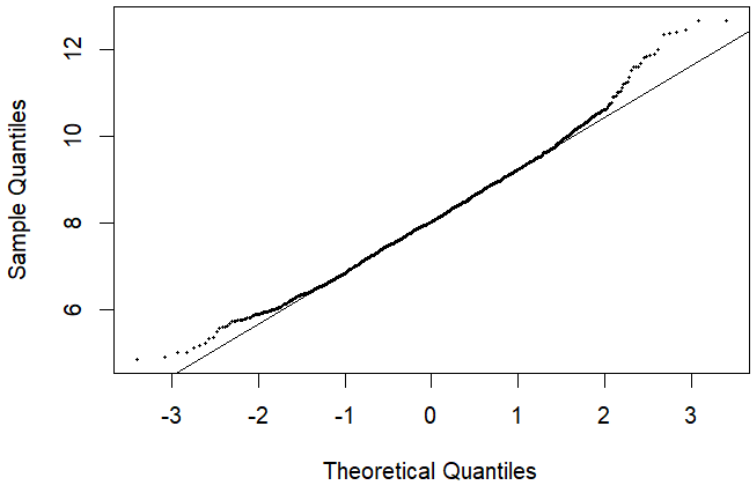


Figure 8.2: Normal Q-Q Plot of Logarithm of the Daily Number of Interruptions

From the plot, we can observe that there are no points on upper right corner that exhibit both high leverage and large standardized residuals.

In accordance with the post-shrinkage method, the independent variables that do not have a significant impact on the response variable are regarded as the nuisance variables. However, they are not to be entirely disregarded. The method employs an adaptive approach by combining the full model with a selected submodel, ensuring that the underlying information from all covariates is taken into consideration. See Equations (1.13) and (1.14).

8.3.2 Submodel Estimation

In order to construct our shrinkage estimators, we utilize various criteria such as AIC, BIC, and LASSO to obtain the corresponding submodels. These methods thus assist in determining the most appropriate prediction equations.

The model that is chosen using the AIC consists of 39 independent variables. The BIC criterion selects a model with only 10 independent variables. The LASSO criterion uses a penalty term to encourage sparsity in the model, effectively shrinking less important variables towards zero and selecting the most relevant variables. It selects a model with 20 independent variables for our data. The selected submodels are given in Tables 8.3, 8.4 and 8.5 respectively.

Table 8.3: Estimates of the Regression Coefficients for the AIC Based Submodel

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
(Intercept)	-0.3886017	1.0541478	0.71245	$R^2 = 0.5843$
X_2	0.0153463	0.0070932	0.030666	Adjusted $R^2 = 0.5729$
X_3	0.0198119	0.0049759	0.0000719	$F = 51.28$
X_4	-0.0109656	0.0064635	0.090003	$RootSqErr = 0.7904$
X_5	0.4323346	0.1017886	0.000023	$pvalue < 2 \times 10^{-16}$
X_6	0.0254886	0.0124398	0.040649	
X_8	-0.0085211	0.0036966	0.021302	
X_9	0.0175848	0.0050791	0.000552	
X_{12}	0.0339535	0.0180309	0.059893	
X_{13}	0.0394245	0.0075529	2.06×10^{-7}	
X_{15}	0.2547198	0.082365	0.002023	

Continued on next page

Table 8.3 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
X_{17}	0.4887778	0.2257501	0.030544	
X_{18}	0.1725939	0.0946096	0.06832	
X_{19}	-0.4179626	0.2271268	0.065945	
X_{22}	1.2597972	0.8119949	0.121008	
X_{24}	-1.3237345	0.8129113	0.103664	
X_{36}	1.164961	0.8083931	0.149781	
X_{37}	0.0324916	0.0084098	0.000117	
X_{38}	0.0343405	0.0097926	0.000468	
X_{39}	0.0368898	0.0110194	0.000836	
X_{40}	0.0343842	0.0079949	0.0000182	
X_{43}	-0.0013326	0.0006201	0.031814	
X_{46}	-0.0006236	0.0003411	0.06771	
X_{47}	0.0073935	0.0026412	0.00519	
X_{49}	0.0117353	0.0025265	0.00000372	
X_{50}	0.0137271	0.0031889	0.0000179	
X_{51}	0.0078807	0.0018214	0.0000162	
X_{52}	0.0066777	0.0014103	0.00000241	
X_{56}	0.0053601	0.0022598	0.017829	
X_{57}	-0.0302398	0.0109005	0.005607	
X_{59}	-0.0325382	0.0102076	0.001465	
X_{60}	-0.0462373	0.0135544	0.000665	
X_{61}	-0.0572511	0.0176298	0.001192	

Continued on next page

Table 8.3 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
X_{63}	-0.1442452	0.0391693	0.000239	
X_{64}	-0.2907236	0.0610766	0.00000213	
X_{67}	0.6553341	0.1449551	0.00000667	
X_{68}	0.9398147	0.1425089	5.99×10^{-11}	
X_{69}	0.774191	0.2061091	0.000179	
X_{70}	1.0798939	0.1170179	$< 2 \times 10^{-16}$	
X_{71}	1.5509134	0.1333018	$< 2 \times 10^{-16}$	

Table 8.4: Estimates of the Regression Coefficients for the BIC Based Submodel

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	P value	Statistics
(Intercept)	5.5667097	0.1037673	$< 2 \times 10^{-16}$	$R^2 = 0.5543$
X_3	0.0274162	0.0014814	$< 2 \times 10^{-16}$	Adjusted $R^2 = 0.5512$
X_5	0.6185568	0.0856295	8.14×10^{-13}	$F = 180.6$
X_{13}	0.05062	0.004026	$< 2 \times 10^{-16}$	$RootSqErr = 0.8102$
X_{15}	0.2599254	0.0706705	0.000244	$pvalue < 2 \times 10^{-16}$
X_{52}	0.0022943	0.0006245	0.000248	
X_{67}	0.7601916	0.1457171	0.000000208	
X_{68}	0.9141751	0.1431418	2.28×10^{-10}	
X_{69}	0.8507536	0.2055544	0.0000369	
X_{70}	1.2717594	0.1104558	$< 2 \times 10^{-16}$	
X_{71}	1.6987975	0.1254119	$< 2 \times 10^{-16}$	

Table 8.5: Estimates of the Regression Coefficients for the LASSO Based Submodel

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
(Intercept)	-0.3309	1.16	0.77555	$R^2 = 0.4413$
X_3	0.028	0.002831	$< 2 \times 10^{-16}$	Adjusted $R^2 = 0.4336$
X_7	-0.00005269	0.0002711	0.84591	$F = 56.95$
X_8	-0.00475	0.004006	0.23586	$RootSqErr = 0.9103$
X_9	0.0199	0.005095	0.000098	$pvalue < 2 \times 10^{-16}$
X_{13}	0.0605	0.009982	1.72×10^{-9}	
X_{14}	0.01282	0.006489	0.04834	
X_{42}	0.002077	0.001197	0.08283	
X_{43}	0.003124	0.001269	0.01392	
X_{44}	0.006257	0.001277	0.00000107	
X_{45}	0.001624	0.0008112	0.04551	
X_{46}	0.0003372	0.000443	0.44665	
X_{47}	0.01031	0.002626	0.0000898	
X_{49}	0.002016	0.002716	0.45803	
X_{50}	0.01226	0.002652	0.00000411	
X_{51}	0.00726	0.002476	0.00341	
X_{52}	-0.002799	0.001879	0.13661	
X_{53}	-0.007022	0.001669	0.0000276	
X_{54}	-0.006697	0.002266	0.00317	
X_{55}	-0.003059	0.001939	0.11494	
X_{56}	0.003273	0.002042	0.10915	

8.3.3 Development of Refined Estimators

In the previous section, we have defined several refined estimators, either based on penalty or shrinkage. Here we obtain these estimators for our data. By applying the technique of bootstrap, we will evaluate their performances. To do so, for each bootstrap sample of same size, we construct a training sample to obtain the model and corresponding estimates of the regression coefficients. These are then evaluated on a test sample for their predictive performances. Test data consists of 25% of the entire sample and these observations are randomly chosen. For each training dataset, we fit the full model and the selected submodels, followed by likelihood ratio tests for each pair of the full model and a selected submodel. We then obtain various post-shrinkage estimators based on Equations 1.13 and 1.14. In addition, the model is fitted using five penalization methods, including LASSO, ENET, ALASSO, SCAD, and Ridge The performance of each model is evaluated based on average prediction error. For illustration, Table 8.6 presents prediction errors for just one bootstrap sample, along with values of $\mathscr{Y}$ and $\hat{\mathscr{Y}} = e^{\hat{Y}}$ given in Table 8.7. For brevity, only nine observations are listed, three each corresponding to low, moderate and high values of $\mathscr{Y}$.

Our evaluation is based on 250 bootstrap samples. We outline the steps followed in our approach below.

1. Perform the bootstrap with 250 iterations.

2. Randomly split each bootstrap sample into training and testing sets in a ratio of $3/4 : 1/4$.

3. For each bootstrap iteration,

Table 8.6: Prediction Error of Estimators in a Single Bootstrap Sample

Estimator	Prediction Error
Full Model	0.7571333
Submodel BIC	0.7258863
Shrinkage Estimator BIC	0.7247393
Positive Shrinkage Estimator BIC	0.7247393
Submodel AIC	0.7282156
Shrinkage Estimator AIC	0.7289327
Positive Shrinkage Estimator AIC	0.7289327
Submodel LASSO	0.9069681
Shrinkage Estimator LASSO	0.7503530
Positive Shrinkage Estimator LASSO	0.7503530
LASSO	0.7520862
ENET	0.7353917
ALASSO	0.7349673
SCAD	0.7318220
RIDGE	0.7550037

(a) Fit the full model using the training dataset and obtain the coefficients of all independent variables.

(b) Fit the three submodels (selected previously using AIC, BIC, and LASSO) using the training dataset and obtain the coefficients of the selected variables in each submodel.

(c) Perform the likelihood ratio test on the full model with each of the above three submodels. With the test statistic compute the shrinkage estimators and positive shrinkage estimators for each pair of full model and submodel estimator.

(d) Fit the models with other five penalized methods (LASSO, ENET, ALASSO, SCAD, and Ridge) using the training dataset and store the coefficients.

(e) Estimate the average prediction errors based on the test dataset for the following models:

i. Full model

ii. Three submodels

iii. Three models with shrinkage estimators

iv. Three models with positive shrinkage estimators

v. Five penalized models (LASSO, ENET, ALASSO, SCAD, and Ridge)

4. Calculate the averages and standard deviations of the mean prediction errors over all bootstrap samples.

By taking the averages and standard deviations of respective mean prediction errors, we can evaluate the performance of each model and compare their predictive accuracy, as shown in the columns 2 and 3 of Table 8.8. When analyzing each bootstrap sample, if we focus solely on the top 100 days with the highest count of outages in the testing dataset, the average prediction errors and corresponding standard deviations for different estimators are updated as columns 4 and 5 of the same table.

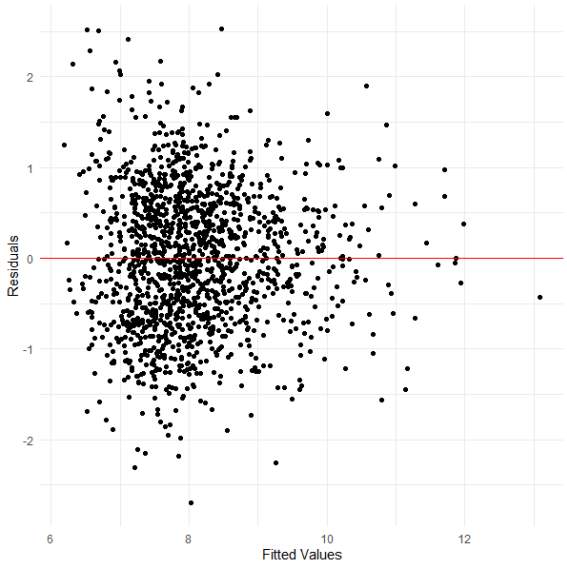
8.3.4 Future Predictions

Models so far obtained have attempted to relate the number of outages to various predictors in current time and they assured us that such associations do exist and improvement can be realized by penalized approach and via shrinkage. The practical problem, however, is the prediction of outages in future, say for next day, in advance, using the past data on predictors. In order to prepare the company a prior and thus minimize the inconvenience to the customers, this is what we actually need. This necessitates modelling of response in terms of lagged variables.

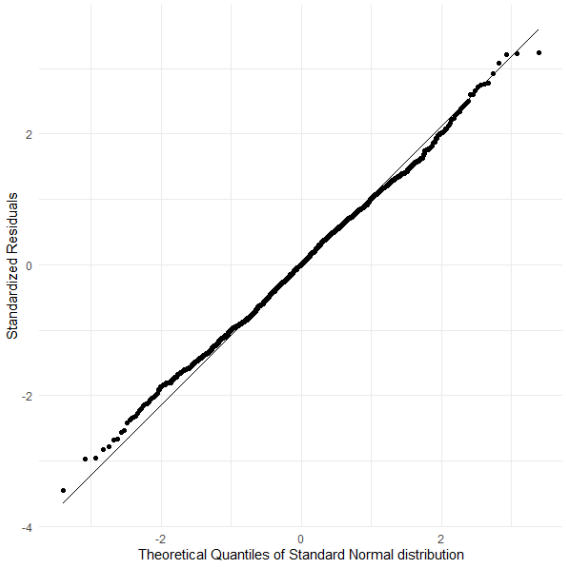
We adopt the same approach as earlier but the predictors are now lagged by one day. Obviously, the estimated coefficients in full model must change and the submodels selected would not be the same as earlier in terms of predictors as well as their coefficients. Further, we must expect somewhat relatively poorer fit and higher average prediction errors. This being the most realistic situation, we present the variables chosen by three criteria in Table 8.9.

As earlier, improved estimation is desired and this is especially welcome given that corresponding full and reduced models will have somewhat poorer fit. Table 8.10 gives us the average prediction errors, computed for the test data, for various approaches. We also give the average prediction errors for 100 worst days of outages.

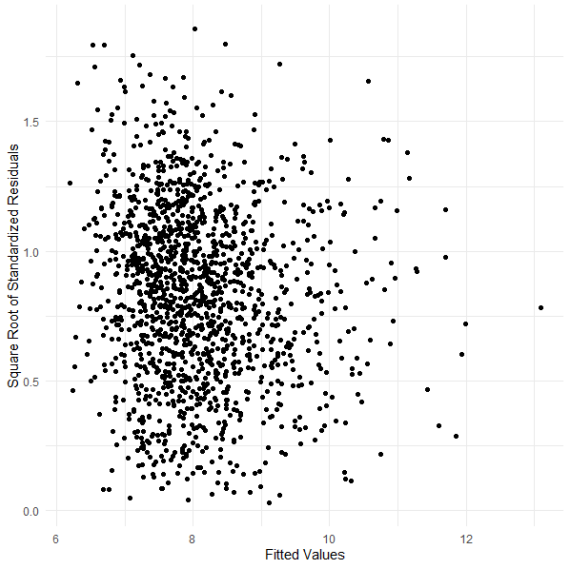
It is clear from Table 8.10 that all shrinkage based methods by using the *AIC* criterion and shrinkage method with *BIC* criterion are quite effective compared to other approaches with respect to average prediction error as well as the average prediction error computed for 100 worst days. Average prediction errors for these winning models are only slightly different from each other. While the average predictor errors are now larger compared to previously obtained models, these observations are consistent with what we observed for the models which related the current response to the current values of the predictors. In Table 8.11 we present the results from one bootstrap simulation and give the actual values of response variable and predicted values.



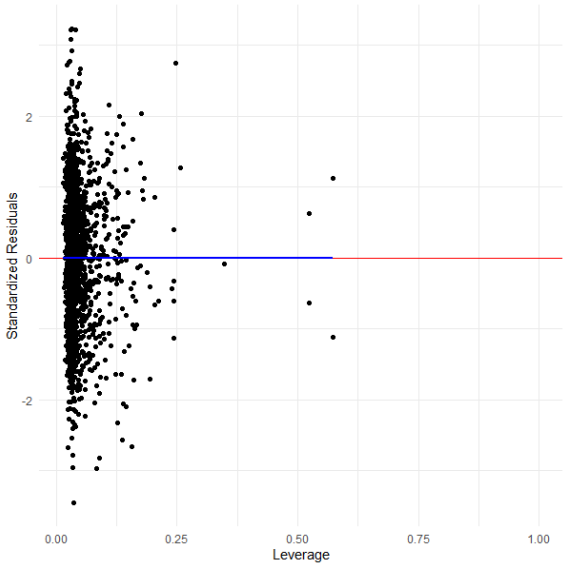
(a) Residuals vs. Fitted Values



(b) Normal Q-Q Plot of Residuals



(c) Examination of Homoscedasticity



(d) Residuals vs. Leverage

Figure 8.3: Full Model Diagnostics Plots

Table 8.7: Predicted Results for Estimators in a Single Bootstrap Sample

Obs	y	$\hat{y}^{FM}$	$\hat{y}^{SM.BIC}$	$\hat{y}^{S.BIC}$	$\hat{y}^{PS.BIC}$	$\hat{y}^{SM.AIC}$	$\hat{y}^{S.AIC}$	$\hat{y}^{PS.AIC}$	$\hat{y}^{SM.LASSO}$	$\hat{y}^{S.LASSO}$	$\hat{y}^{PS.LASSO}$	$\hat{y}^{LASSO}$	$\hat{y}^{ENET}$	$\hat{y}^{ALASSO}$	$\hat{y}^{SCAD}$	$\hat{y}^{RIDGE}$
1	149	869	933	900	900	903	911	903	723	845	845	969	1046	863	918	973
2	206	3246	2177	2668	2668	2864	2780	2864	2685	3153	3153	2349	2686	2239	2309	3048
3	214	746	1132	916	916	714	707	714	952	775	775	1313	1148	1181	1081	907
4	2959	3629	3887	3754	3754	3495	3464	3495	3639	3631	3631	3943	3412	3780	3662	4044
5	2967	3258	3273	3265	3265	3085	3045	3085	4529	3425	3425	3226	3179	3211	3216	3185
6	2999	3323	3350	3336	3336	3554	3611	3554	4093	3430	3430	3208	3413	2958	3309	3439
7	37502	32635	24261	28215	28215	37316	38522	37316	14881	28957	28957	16758	25602	26546	25172	26354
8	40506	36863	43975	40197	40197	44885	47031	44885	16022	32471	32471	32128	47411	46765	45116	49730
9	48150	43429	51916	47405	47405	63218	69108	63218	15251	37032	37032	32635	51490	47709	51241	53767

Table 8.8: Average and Standard Deviation of Prediction Errors for Estimators (All Data and Worst 100 days' Data)

Estimator	All Data		Worst 100 days' Data	
	PE Average	PE s.d	PE Average	PE s.d
Full Model	0.6477945	0.04440908	0.8649341	0.1218069
Submodel BIC	0.6582174	0.04518586	0.9213475	0.1210548
Shrinkage Estimator BIC	0.6312453	0.04219535	0.8625677	0.1165269
Positive Shrinkage Estimator BIC	0.6312453	0.04219535	0.8625677	0.1165269
Submodel AIC	0.6330242	0.04205691	0.8476032	0.1158598
Shrinkage Estimator AIC	0.6323249	0.04231598	0.8461770	0.1165548
Positive Shrinkage Estimator AIC	0.6322360	0.04223741	0.8459589	0.1163747
Submodel LASSO	0.8333164	0.06673248	1.2686779	0.2114287
Shrinkage Estimator LASSO	0.6439578	0.04430009	0.8544253	0.1229643
Positive Shrinkage Estimator LASSO	0.6439578	0.04430009	0.8544253	0.1229643
LASSO	0.6831690	0.05012177	0.9933399	0.1427906
ENET	0.6622813	0.04603949	0.9458174	0.1298107
ALASSO	0.6692853	0.04639950	0.9467351	0.1251344
SCAD	0.6624375	0.04663982	0.9174038	0.1229535
RIDGE	0.6417907	0.04345950	0.8844112	0.1201845

Table 8.9: Selected Variables for Submodels of Future Predictions

Criterion	Variables Selected
BIC	$X_1, X_3, X_4, X_5, X_9, X_{17}, X_{33}, X_{41}, X_{46}, X_{47}, X_{49}, X_{50}, X_{51}, X_{52}, X_{61}, X_{64}, X_{68}, X_{70}, X_{71}$
AIC	$X_2, X_3, X_4, X_5, X_7, X_9, X_{14}, X_{16}, X_{17}, X_{25}, X_{26}, X_{30}, X_{32}, X_{33}, X_{37}, X_{38}, X_{39}, X_{40}, X_{41}, X_{43}, X_{46}, X_{47}, X_{49}, X_{50}, X_{51}, X_{52}, X_{56}, X_{57}, X_{59}, X_{60}, X_{61}, X_{63}, X_{64}, X_{68}, X_{70}, X_{71}$
LASSO	$X_3, X_4, X_7, X_{14}, X_{42}, X_{43}, X_{44}, X_{45}, X_{46}, X_{47}, X_{49}, X_{50}, X_{52}, X_{53}, X_{54}, X_{55}, X_{56}$

Table 8.10: Average and Standard Deviation of Prediction Errors for Future (One-Day-Forward) Predictive Models (All Data and Worst 100 days' Data)

Estimator	All Data		Worst 100 days' Data	
	PE Average	PE s.d.	PE Average	PE s.d.
Full Model	0.9404684	0.07821941	1.584245	0.2621179
Submodel BIC	0.9245679	0.07694673	1.614551	0.2635939
Shrinkage Estimator BIC	0.9121116	0.07522929	1.578550	0.2592123
Positive Shrinkage Estimator BIC	0.9121116	0.07522929	1.578550	0.2592123
Submodel AIC	0.9144040	0.07440663	1.557241	0.2581588
Shrinkage Estimator AIC	0.9148878	0.07460741	1.556348	0.2569069
Positive Shrinkage Estimator AIC	0.9146506	0.07468312	1.556009	0.2570214
Submodel LASSO	1.0542989	0.09158692	1.882316	0.3277655
Shrinkage Estimator LASSO	0.9300443	0.07655863	1.586973	0.2655936
Positive Shrinkage Estimator LASSO	0.9300443	0.07655863	1.586973	0.2655936
LASSO	0.9731440	0.08809294	1.775802	0.3086070
ENET	0.9559625	0.08493129	1.735077	0.3007429
ALASSO	0.9646395	0.08473409	1.729453	0.2979533
SCAD	0.9597182	0.08252438	1.698725	0.2891473
RIDGE	0.9289217	0.07830412	1.623942	0.2777160

Table 8.11: Predicted Results for Future (One-Day-Forward) Predictive Models in a Single Bootstrap Sample

Obs	y	$\hat{y}_{FM}$	$\hat{y}_{SM.BIC}$	$\hat{y}_{S.BIC}$	$\hat{y}_{PS.BIC}$	$\hat{y}_{SM.AIC}$	$\hat{y}_{S.AIC}$	$\hat{y}_{PS.AIC}$	$\hat{y}_{SM.LASSO}$	$\hat{y}_{S.LASSO}$	$\hat{y}_{PS.LASSO}$	$\hat{y}_{LASSO}$	$\hat{y}_{ENET}$	$\hat{y}_{ALASSO}$	$\hat{y}_{SCAD}$	$\hat{y}_{RIDGE}$
1	135	1569	1719	1649	1649	1559	1562	1562	2229	1704	1704	2026	1913	1908	2084	1629
2	167	812	954	886	886	861	847	847	757	798	798	1031	967	910	921	764
3	305	1385	1625	1510	1510	1307	1330	1330	1226	1346	1346	1649	1591	1510	1645	1393
4	3105	2264	2085	2166	2166	2673	2545	2545	1751	2131	2131	1986	2004	1727	1863	2046
5	3156	1789	1537	1647	1647	1725	1743	1743	2180	1874	1874	2160	2033	1885	1881	1699
6	3171	2745	2817	2784	2784	2503	2572	2572	2987	2800	2800	2790	2864	2772	2880	2960
7	40510	20576	14057	16741	16741	16037	17255	17255	8075	16509	16509	11174	11822	13723	11452	17276
8	40891	3491	4750	4125	4125	3989	3836	3836	5882	3948	3948	5463	5373	6142	5600	4790
9	41525	19748	13801	16266	16266	14375	15781	15781	12306	17667	17667	13031	14212	16545	13532	19968

8.4 Copula Transformation Based Model

Following Bahuguna and Khattree (2020), we will conduct the analysis of our data after transformation and on the copula transformed data we will fit all the previous models. The predictions can be back-transformed as shown in above diagram. The transformation is applied on the response variable and the 46 numerical independent variables. Subsequently, we conduct regression analysis on these 46 transformed variables, along with other original 27 dummy or binary variables, for the transformed response variables. As earlier, all the previous approached will be adopted but this time on the copula-transformed data.

For transforming predictions back to the original data, we will apply the interpolation techniques suggested by Bahuguna and Khattree. For this interpolation, we simulate 2000 draws from the fitted copula distribution. These simulated draws are subsequently employed to transform the prediction results back to the original scale. Details are given in above mentioned article.

8.4.1 Full Model and Submodels for the Transformed Data

Let the transformed variables be denoted by W_i, while the remaining untransformed predictors retain their original notation as X_i. Following the same approach as for the analysis conducted on the original data, we first fit a full regression model. The results of the regression analysis are presented in the Table 8.12.

Table 8.12: Estimates of the Regression Coefficients for the Full Model on Copula-Transformed Data

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
(Intercept)	-0.158516	0.17839	0.374376	$R^2 = 0.555$

Continued on next page

Table 8.12 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
W_1	0.204954	0.077023	0.007882	Adjusted $R^2 = 0.5316$
W_2	0.132963	0.087086	0.127036	$F = 23.73$
W_3	0.151046	0.082741	0.068137	$RootSqErr = 0.6814$
W_4	-0.166578	0.092056	0.070587	$pvalue < 2 \times 10^{-16}$
W_5	0.314271	0.062271	5.09×10^{-7}	
W_6	0.084999	0.039185	0.030241	
W_7	0.00588	0.019243	0.759983	
W_8	-0.035968	0.057407	0.531065	
W_9	0.05395	0.074249	0.467593	
W_{10}	0.06312	0.030894	0.041229	
W_{11}	-0.138841	0.084457	0.100417	
W_{12}	0.228826	0.093337	0.014344	
W_{13}	0.094778	0.055243	0.086447	
W_{14}	0.035567	0.046591	0.445361	
W_{15}	0.114181	0.05694	0.045125	
W_{16}	0.058059	0.067256	0.388147	
W_{17}	-0.02301	0.085937	0.788933	
W_{18}	0.11397	0.093818	0.22465	
W_{19}	0.106335	0.084738	0.209735	
W_{20}	-0.088829	0.055505	0.109741	
W_{21}	-0.120921	0.072881	0.097311	
W_{22}	-0.043734	0.100022	0.661999	

Continued on next page

Table 8.12 – continued from previous page

Variable	$\hat{\beta}_i$	$S_{\hat{\beta}_i}$	P value	Statistics
W_{23}	-0.00588	0.054722	0.914445	
W_{24}	-0.08539	0.09506	0.369193	
W_{25}	0.133336	0.057735	0.021065	
W_{26}	0.091756	0.072493	0.205826	
W_{27}	0.201351	0.113695	0.076785	
W_{28}	0.280094	0.134295	0.037192	
W_{29}	0.184172	0.124645	0.139749	
W_{30}	0.098836	0.061708	0.109455	
W_{31}	-0.006723	0.057666	0.907206	
W_{32}	-0.034515	0.072073	0.632092	
W_{33}	0.062702	0.115585	0.587582	
W_{34}	0.084896	0.091719	0.354807	
W_{35}	-0.131098	0.064182	0.04128	
W_{36}	0.004218	0.056981	0.941003	
W_{37}	-0.058226	0.061821	0.346432	
W_{38}	-0.08736	0.076505	0.253701	
W_{39}	-0.089138	0.079306	0.261218	
W_{40}	0.005667	0.077751	0.941908	
W_{41}	-0.150096	0.076075	0.048694	
W_{42}	-0.096618	0.044854	0.031409	
W_{43}	-0.019401	0.039968	0.627464	
W_{44}	-0.063243	0.035684	0.076567	

Continued on next page

Table 8.12 – continued from previous page

Variable	$\hat{\beta}_i$	$S_{\hat{\beta}_i}$	P value	Statistics
W_{45}	-0.081839	0.066654	0.219725	
W_{46}	0.016881	0.043055	0.695058	
X_{15}	0.225342	0.082801	0.00658	
X_{16}	0.049593	0.255669	0.846225	
X_{17}	0.112102	0.222735	0.614835	
X_{18}	0.057542	0.091371	0.528954	
X_{19}	-0.121914	0.222853	0.584426	
X_{20}	-0.127048	0.180077	0.480605	
X_{21}	0.12147	0.141146	0.389606	
X_{22}	0.976298	0.741046	0.187902	
X_{23}	-0.163633	0.247567	0.508744	
X_{24}	-0.945779	0.768792	0.218825	
X_{25}	0.124238	0.21109	0.556258	
X_{26}	-0.091358	0.143935	0.525718	
X_{27}	0.006749	0.501941	0.989274	
X_{28}	0.067234	0.139672	0.63033	
X_{29}	0.068221	0.221156	0.757769	
X_{30}	0.007335	0.322748	0.981872	
X_{31}	-0.440775	0.518885	0.395769	
X_{32}	-0.052212	0.116721	0.654715	
X_{33}	-0.227244	0.188966	0.229349	
X_{34}	0.086631	0.160935	0.590459	

Continued on next page

Table 8.12 – continued from previous page

Variable	$\hat{\beta}_i$	$\hat{S_{\beta_i}}$	**P value**	**Statistics**
X_{35}	-0.151252	0.182432	0.407199	
X_{36}	1.06347	0.725289	0.142801	
X_{67}	0.653768	0.128307	3.96×10^{-7}	
X_{68}	0.718192	0.128521	2.76×10^{-8}	
X_{69}	0.647709	0.181328	0.000366	
X_{70}	0.848135	0.104235	8.92×10^{-16}	
X_{71}	1.155344	0.118925	$< 2 \times 10^{-16}$	

The variables chosen by BIC, AIC and LASSO criteria are presented in Table 8.13.

Table 8.13: Selected Variables for Submodels on Copula-Transformed Data

Criterion	Variables Selected
BIC	$W_1, W_5, W_6, W_{10}, W_{20}, W_{28}, W_{29}, W_{42}, W_{36}, X_{15}, X_{67}, X_{68}, X_{69}, X_{70}, X_{71}$
AIC	$W_1, W_2, W_3, W_4, W_5, W_6, W_{10}, W_{11}, W_{12}, W_{13}, W_{15}, W_{16}, W_{18}, W_{20}, W_{21}, W_{25}, W_{27}$, $W_{28}, W_{29}, W_{30}, W_{35}, W_{38}, W_{39}, W_{41}, W_{42}, W_{44}, X_{15}, X_{23}, X_{67}, X_{68}, X_{69}, X_{70}, X_{71}$
LASSO	$W_2, W_3, W_5, W_6, W_{13}, W_{22}, W_{42}, W_{44}$

8.4.2 Computation of Average Prediction Errors

Since predictions are obtained for the Copula-transformed responses, we must take an additional step to re-transform these predictions to the original scale. This is the $\mathscr{N} \longrightarrow \mathscr{U} \longrightarrow \mathscr{D}$ step of back transformation. For every $i = 1, ..., n, \mathscr{N}_i \longrightarrow \mathscr{U}_i$ step is straight forward and then by simulating 2000 observations from the Copula distribution,

we interlace the two sets of observations in increasing order and then for each of $\mathscr{U}_i$ we identify the immediate lower and upper values from the simulated observations. Let them be $\mathscr{U}_{i,lower}$ and $\mathscr{U}_{i,upper}$. Accordingly, by $\mathscr{U} \longrightarrow \mathscr{D}$ step, one can obtain limits on $\hat{F}^{-1}(u_i)$ as

$$\hat{F}^{-1}(u_{i,lower}) \leq \hat{F}^{-1}(u_i) \leq \hat{F}^{-1}(u_{i,upper}),$$

where $\hat{F}^{-1}$ is the empirical distribution function of the data. Thus we approximate the prediction $\hat{F}^{-1}(u_i)$ as

$$\hat{y}_i = \hat{F}^{-1}(u_i) = \frac{\hat{F}^{-1}(u_{i,lower}) + \hat{F}^{-1}(u_{i,upper})}{2}$$

Table 8.14: Example of Transforming Copula-Transformed Data Back to the Original Scale

$\hat{y}_i^*$	u_i	$u_{i,lower}$	$\hat{F}^{-1}(u_{i,lower})$	$u_{i,upper}$	$\hat{F}^{-1}(u_{i,upper})$	$\hat{F}^{-1}(u_i) = \hat{y}_i$
-1.6757	0.0469	0.0468	462.9167	0.0494	478.4278	470.6723
-1.5875	0.0562	0.0559	512.1258	0.0563	515.768	513.9469
-1.5649	0.0588	0.0578	526.6234	0.0593	536.6676	531.6455
-1.5433	0.0614	0.0612	546.6164	0.0615	547.9147	547.2655
-1.5347	0.0624	0.0621	550.8367	0.0628	554.7244	552.7805

Hence, for each set of predicted values, we convert them back to the original scale. The Table 8.14 presents data for an example, the column $\hat{y}_i^*$ shows the prediction results from the full model estimator. By applying the CDF, we transform these values into uniform values, which are displayed in the column u_i. By interlacing the uniform values in ascending order with the simulation list, we identify their positions and corresponding

positions in the original scale list. The column $\hat{F}^{-1}(u_{i,lower})$ and column $\hat{F}^{-1}(u_{i,upper})$ depict the immediate lower and upper value in the original scale. The column $\hat{F}^{-1}(u_i)$ represents the mean of these two values, which serves as the prediction result that has been converted back to the original scale $\hat{y}_i$.

Table 8.15: Average and Standard Deviation of Prediction Errors for Estimators on Copula-Transformed Data (All Data and Worst 100 days' Data)

Estimator	All Data		Worst 100 days' Data	
	PE Average	PE s.d.	PE Average	PE s.d.
Full Model	0.6765421	0.05158322	0.9780245	0.1352269
Submodel BIC	1.1352706	0.15227586	1.4708888	0.2560801
Shrinkage Estimator BIC	0.7774494	0.08070030	1.0353053	0.1706364
Positive Shrinkage Estimator BIC	0.7774494	0.08070030	1.0353053	0.1706364
Submodel AIC	1.4533780	0.21541327	1.9961574	0.3269358
Shrinkage Estimator AIC	1.0850871	0.23611706	1.5246786	0.3454253
Positive Shrinkage Estimator AIC	1.0777394	0.21681108	1.5181878	0.3324443
Submodel LASSO	0.8124976	0.06575758	1.3034233	0.1874719
Shrinkage Estimator LASSO	0.6675341	0.04983899	0.9619997	0.1291368
Positive Shrinkage Estimator LASSO	0.6675341	0.04983899	0.9619997	0.1291368
LASSO	0.7300759	0.05868800	1.1453310	0.1557111
ENET	0.6940998	0.05279401	1.0610378	0.1385513
ALASSO	0.7046229	0.05112499	1.0701990	0.1372028
SCAD	0.6886609	0.05258059	1.0267939	0.1332745
RIDGE	0.6631083	0.04756883	0.9687484	0.1218036

In order to compare the average prediction errors on the log scale, we take the logarithm of both the prediction results and the actual response values. Subsequently, we calculate the prediction errors for each bootstrap sample. The Table 8.15 columns 2 and 3 present the average prediction errors using this approach. The columns 4 and 5 of the

same table display the average and standard deviation of prediction errors for different estimators for the bootstrap sample consisting of the top 100 days with the highest count of outages.

8.4.3 Future Predictions Using Copula-Transformed Approach

Table 8.16: Selected Variables for Submodels of Future Predictions on Copula-Transformed Data

Criterion	Variables Selected
BIC	$W_1, W_5, W_7, W_{19}, W_{44}, X_{68}, X_{70}, X_{71}$
AIC	$W_1, W_4, W_5, W_7, W_8, W_9, W_{11}, W_{12}, W_{14}, W_{15}, W_{17}, W_{18}, W_{19}, W_{25}, W_{30}, W_{38}, W_{39}, W_{42}, W_{44}, X_{16}, X_{22}, X_{24}, X_{25}, X_{26}, X_{30}, X_{33}, X_{68}, X_{70}, X_{71}$
LASSO	$W_1, W_3, W_5, W_{13}, W_{42}, W_{43}$

As earlier, we consider an analysis that utilizes the independent variables to forecast the number of outages for the following day. To achieve this, we employ the same dataset but employ the next day's outage value as the response variable. We then perform the regression using the full model estimator and apply BIC, AIC, LASSO to select submodels, as shown in Table 8.16.

Using the same approach as earlier, we evaluate the performances of various approached here as well. The average prediction errors are listed in Table 8.17 columns 2 and 3. The comparison result for the testing dataset of the top 100 days with the highest count of outages is in the same table columns 4 and 5. Compared to results from previous analyses on the response variable $\mathbf{Y}$ shown in Tables 8.8 and 8.10, the Copula-transformed

Table 8.17: Average and Standard Deviation of Prediction Errors for Future (One-Day-Forward) Predictive Model on Copula-Transformed Data (All Data and Worst 100 days' Data)

Estimator	All Data		Worst 100 days' Data	
	PE Average	PE s.d.	PE Average	PE s.d
Full Model	0.9348529	0.07491297	1.600254	0.2491508
Submodel BIC	0.9437523	0.08346633	1.698721	0.2720802
Shrinkage Estimator BIC	0.9125156	0.07830735	1.613821	0.2597616
Positive Shrinkage Estimator BIC	0.9125156	0.07830735	1.613821	0.2597616
Submodel AIC	0.9107263	0.07618440	1.597390	0.2548192
Shrinkage Estimator AIC	0.9084816	0.07545964	1.585500	0.2556990
Positive Shrinkage Estimator AIC	0.9085226	0.07541713	1.585631	0.2554379
Submodel LASSO	1.0182377	0.08933894	1.833601	0.2937266
Shrinkage Estimator LASSO	0.9216676	0.07691704	1.610214	0.2562074
Positive Shrinkage Estimator LASSO	0.9216676	0.07691704	1.610214	0.2562074
LASSO	0.9947908	0.09162398	1.816666	0.2864837
ENET	0.9718344	0.08585098	1.771969	0.2809249
ALASSO	0.9689961	0.08636857	1.667906	0.3048315
SCAD	0.9539474	0.08139597	1.671269	0.2834683
RIDGE	0.9196266	0.07550722	1.606128	0.2513853

model yields slightly larger prediction errors for present model and exhibits very similar prediction errors for one-day-forward forecasts.

It is also of interest to evaluate if Copula transformation helps improve predictions for our data. Table 8.18 presents various test statistics for the corresponding full model where (i) data on $\mathscr{Y}$ are used (ii) data on $\mathbf{Y} = log(\mathscr{Y})$ are used and (iii) Copula-transformed data are used. The model for the Present-day prediction using Copula-transformed data yields a lower R^2 (=0.5550) compared to the other two models. For the models of one-day-forward prediction, as we have already seen that the data on $\mathscr{Y}$ was highly skewed and thus resulted in very low R^2 (=0.2031). The other two models

Table 8.18: Statistics of Full Models for Future (One-Day-Forward) Prediction on Different Response Variables

Type	Response	R^2	Adjusted R^2	RootSqErr	F statistic
Present Day	Daily Interruption	0.5843	0.5625	14160	26.75
	Logarithm of Daily Interruption	0.5891	0.5675	0.7954	27.28
	Copula-transformed Daily Interruption	0.5550	0.5316	0.6814	23.73
Future Prediction (One-Day-Forward)	Daily Interruption	0.2031	0.1612	19610	4.846
	Logarithm of Daily Interruption	0.4021	0.3707	0.9588	12.79
	Copula-Transformed Daily Interruption	0.3941	0.3622	0.7952	12.37

have very comparable R^2 values (0.4021 and 0.3941 respectively). This suggests that Copula transformation has not improved the full model substantially. The same can be seen for other statistics such as the F-statistic.

8.5 The Analysis of Entire Data

So far, we have used the training data to arrive at various models and evaluated them on the test data. For completeness, it is also of interest to analyze the entire data and look at the prediction errors. Clearly since the models use all observations (combining training data and test data) and then compute prediction errors for the very same observations that were used to arrive at the model, the situation may look better than it actually will be when a yet-unobserved future value is to be predicted. These one time analyses cannot be used for statistical comparisons of various approaches. Nonetheless, it is still of interest to fit these models and looks at the prediction errors. These are given in Table 8.19. As one observes, in most cases shrinkage does help. The prediction error values are smaller, except for the last column corresponding to predictions for the next day and for 100 worst days. Smaller prediction errors are to be expected as data and models

Table 8.19: Prediction Errors for Two Scenarios (Using Present Day Data and One-Day in the Past Data) for the Whole Dataset (Training and Test Combined)

Estimator	Present Day Data		One-Day in the Past Data	
	All Data PE	Worst 100 Days PE	All Data PE	Worst 100 Days PE
Full Model	0.6006267	0.7834790	0.8727491	3.074309
Submodel BIC	0.6515199	1.0598402	0.9085831	3.340174
Shrinkage Estimator BIC	0.6135665	0.8923914	0.8996807	3.296670
Positive Shrinkage Estimator BIC	0.6135665	0.8923914	0.8996807	3.296670
Submodel AIC	0.6077234	0.8050504	0.8820751	3.130226
Shrinkage Estimator AIC	0.6237204	0.8481040	0.9173935	3.255268
Positive Shrinkage Estimator AIC	0.6077234	0.8050504	0.8820751	3.130226
Submodel LASSO	0.8166715	2.3560929	1.0351889	4.411524
Shrinkage Estimator LASSO	0.6032996	0.8471173	0.8800761	3.238107
Positive Shrinkage Estimator LASSO	0.6032996	0.8471173	0.8800761	3.238107
LASSO	0.6617996	1.3216146	0.9340344	3.738566
ENET	0.6378105	1.1286216	0.9290832	3.696251
ALASSO	0.6576867	1.1023822	0.9385295	3.603580
SCAD	0.6597183	1.1226969	0.9431967	3.667746
RIDGE	0.6137524	0.9327309	0.8918282	3.376004

together are self-validating in that the corresponding models were optimal for the given data and thus their best performances will naturally be seen for that very data.

8.6 Concluding Remarks

Based on the extensive analyses of the data, few facts come up which we summarize below. The original data on $\mathscr{Y}$ was highly skewed with respect to the response variables, $\mathscr{Y}$, the daily interruptions. Thus a transformation of data was called for. The logarithmic transformation as well as Copula transformation both considerably improved the full model. However performance of these two models were comparable. There however can be situation when transformation by Copula can result in superior performance.

Uniformly, the model selection method *AIC* appears to work favorably when it comes to smaller mean prediction errors. There are instances when *BIC* also gave the

comparable performance. However *LASSO* did not seem to give that level of performance. This is true for both sets of analyses, when we use the current values of predictors as well as when lagged predictors are used. This is true for the log-transformed data as well as the Copula-transformed data.

Generally, shrinkage based models outperform the penalized regression models, in terms of average prediction errors. The performances of all shrinkage methods under *AIC* to select the variables were essentially interchangeable. The dataset is large enough here and thus positive shrinkage was perhaps not needed because $1-(p_2-2)\tau_n^{-1}$ was greater than zero, in most of the bootstrapped cases.

For the worst 100 days with highest number of interruptions, as expected, the average prediction errors were larger. Nonetheless, the patterns in the relative comparisons of various approaches did not change. The conclusions still remain the same.

One may ask, why not just use a submodel while ignoring all the statistically insignificant variables. There are two legitimate reasons for not doing so: First, there is bound to be high level of multicollinearity in such data with 73 predictors. Thus the statistical insignificance of various variables only indicates lack of significance of these while all the other statistically significant variables are retained in the model. Secondly, the dataset is also large enough that despite 73 predictors in the model, it leaves a very large degree of freedom for error. Thus R^2 may be low while the F statistic for any model will be highly statistically significant. That leaves much ambiguity in the interpretation of corresponding submodels. Consequently, it is better to still incorporate the information from these other variables. The shrinkage based estimators do exactly that.

APPENDIX A

SOURCE CODE OF CHAPTER TWO

A.1 Basic Descriptive Analysis

```
# Import Libraries
library(dplyr)
library(zoo)
library(psych)
library(ggplot2)
library(moments)
library(gridExtra)
library(tidyverse)
library(corrplot)
library(psychTools)
library(XICOR)
library ('lmtest')
library ('leaps')
library(stats)
library (glmnet)
library(ncvreg)
library(boot)
library(caTools)
library(fastDummies)
library(car)

# Read the CSV files
Bitcoin <- read.csv("BTC-USD.csv", header = TRUE)
Ethereum <- read.csv("ETH-USD.csv", header = TRUE)
EthereumClassic <- read.csv("ETC-USD.csv", header = TRUE)
TetherUSDt <- read.csv("USDT-USD.csv", header = TRUE)
XRP <- read.csv("XRP-USD.csv", header = TRUE)
Cardano <- read.csv("ADA-USD.csv", header = TRUE)
Dogecoin <- read.csv("DOGE-USD.csv", header = TRUE)

# Preprocess data
Bitcoin$Coin <-"Bitcoin"
# Convert the Date column to actual date format
Bitcoin$Date <- as.Date(Bitcoin$Date,format = "%Y-%m-%d")
# Sort the dataframe by the Date column
Bitcoin <- arrange(Bitcoin, Date)
# Compute the daily returns
Bitcoin <- Bitcoin %>%
```

```
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))

# Compute the moving weekly average
Bitcoin$MAvgClose <- rollmean(Bitcoin$Close, k = 7, align = "right", fill = NA)
Bitcoin$MAvgReturn <- rollmean(Bitcoin$Daily_Return, k = 7, align = "right",
fill = NA)
day0Price <- Bitcoin[which.min(Bitcoin$Date), "Close"]
Bitcoin$StdPrice <- (Bitcoin$Close*100)/day0Price
Bitcoin$MAvgStdPrice <- rollmean(Bitcoin$StdPrice, k = 7, align = "right",
fill = NA)

Ethereum$Coin <- "Ethereum"
Ethereum$Date <- as.Date(Ethereum$Date,format = "%Y-%m-%d")
Ethereum <- arrange(Ethereum, Date)
Ethereum <- Ethereum %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
Ethereum$MAvgClose <- rollmean(Ethereum$Close, k = 7,align = "right", fill = NA)
Ethereum$MAvgReturn <- rollmean(Ethereum$Daily_Return, k = 7, align = "right",
fill = NA)
Ethereum$StdPrice <- (Ethereum$Close*100)/Ethereum[which.min(Ethereum$Date),
"Close"]
Ethereum$MAvgStdPrice <- rollmean(Ethereum$StdPrice, k = 7, align = "right",
fill = NA)

EthereumClassic$Coin <- "EthereumClassic"
EthereumClassic$Date <- as.Date(EthereumClassic$Date,format = "%Y-%m-%d")
EthereumClassic <- arrange(EthereumClassic, Date)
EthereumClassic <- EthereumClassic %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
EthereumClassic$MAvgClose <- rollmean(EthereumClassic$Close, k = 7,
align = "right", fill = NA)
EthereumClassic$MAvgReturn <- rollmean(EthereumClassic$Daily_Return, k = 7,
align = "right", fill = NA)
EthereumClassic$StdPrice <-
(EthereumClassic$Close*100)/EthereumClassic[which.min(EthereumClassic$Date),
"Close"]
EthereumClassic$MAvgStdPrice <- rollmean(EthereumClassic$StdPrice, k = 7,
align = "right", fill = NA)

TetherUSDt$Coin <- "TetherUSDt"
```

```
TetherUSDt$Date <- as.Date(TetherUSDt$Date,format = "%Y-%m-%d")
TetherUSDt <- arrange(TetherUSDt, Date)
TetherUSDt <- TetherUSDt %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
TetherUSDt$MAvgClose <- rollmean(TetherUSDt$Close, k = 7, align = "right",
fill = NA)
TetherUSDt$MAvgReturn <- rollmean(TetherUSDt$Daily_Return, k = 7, align = "right",
fill = NA)
TetherUSDt$StdPrice <- (TetherUSDt$Close*100)/TetherUSDt[which.min(TetherUSDt$Date),
"Close"]
TetherUSDt$MAvgStdPrice <- rollmean(TetherUSDt$StdPrice, k = 7, align = "right",
fill = NA)

XRP$Coin <- "XRP"
XRP$Date <- as.Date(XRP$Date,format = "%Y-%m-%d")
XRP <- arrange(XRP, Date)
XRP <- XRP %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
XRP$MAvgClose <- rollmean(XRP$Close, k = 7, align = "right", fill = NA)
XRP$MAvgReturn <- rollmean(XRP$Daily_Return, k = 7, align = "right", fill = NA)
XRP$StdPrice <- (XRP$Close*100)/XRP[which.min(XRP$Date), "Close"]
XRP$MAvgStdPrice <- rollmean(XRP$StdPrice, k = 7, align = "right", fill = NA)

Cardano$Coin <- "Cardano"
Cardano$Date <- as.Date(Cardano$Date,format = "%Y-%m-%d")
Cardano <- arrange(Cardano, Date)
Cardano <- Cardano %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
Cardano$MAvgClose <- rollmean(Cardano$Close, k = 7, align = "right", fill = NA)
Cardano$MAvgReturn <- rollmean(Cardano$Daily_Return, k = 7, align = "right",
fill = NA)
Cardano$StdPrice <- (Cardano$Close*100)/Cardano[which.min(Cardano$Date), "Close"]
Cardano$MAvgStdPrice <- rollmean(Cardano$StdPrice, k = 7, align = "right",
fill = NA)

Dogecoin$Coin <- "Dogecoin"
Dogecoin$Date <- as.Date(Dogecoin$Date,format = "%Y-%m-%d")
Dogecoin <- arrange(Dogecoin, Date)
Dogecoin <- Dogecoin %>%
  mutate(Daily_Return = c(NA, diff(Close)/lag(Close)[-1]))
```

```
Dogecoin$MAvgClose <- rollmean(Dogecoin$Close, k = 7, align = "right",
fill = NA)
Dogecoin$MAvgReturn <- rollmean(Dogecoin$Daily_Return, k = 7, align = "right",
fill = NA)
Dogecoin$StdPrice <- (Dogecoin$Close*100)/Dogecoin[which.min(Dogecoin$Date),
"Close"]
Dogecoin$MAvgStdPrice <- rollmean(Dogecoin$StdPrice, k = 7, align = "right",
fill = NA)

# Combine the files
df <- rbind(Bitcoin, Ethereum,EthereumClassic,TetherUSDt,XRP,Cardano,Dogecoin)
head(df)
summary(df)

# Compute summary statistics for each coin
summary_stats <- df %>%
  group_by(Coin) %>%
  summarise(min = min(Daily_Return, na.rm = TRUE),
            q1 = quantile(Daily_Return, probs = 0.25, na.rm = TRUE),
            median = median(Daily_Return, na.rm = TRUE),
            mean = mean(Daily_Return, na.rm = TRUE),
            q3 = quantile(Daily_Return, probs = 0.75, na.rm = TRUE),
            max = max(Daily_Return, na.rm = TRUE),
            sd = sd(Daily_Return, na.rm = TRUE),
            skewness = psych::skew(Daily_Return, na.rm = TRUE),
            kurtosis = moments::kurtosis(Daily_Return, na.rm = TRUE))

# Print summary statistics
print(summary_stats)
Dogecoin[which.max(Dogecoin$Daily_Return), ]
Dogecoin[(Dogecoin$Date >'2021-01-24')&(Dogecoin$Date <'2021-01-31'), -6]

# Calculate the largest values for each type
df_max <- df %>%
  group_by(Coin) %>%
  top_n(1, Daily_Return) %>%
  ungroup()
print(df_max)

# Plot boxplot for each type
```

```
ggplot(df, aes(x = Coin, y = Daily_Return)) +
  geom_boxplot() +
  geom_text(data = df_max, aes(label = round(Daily_Return, 2)),
            position = position_dodge(width = 0.5), vjust = -0.5, color = "red") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black")) + # Add y-axis line
  labs(x = "Cryptocurrency", y = "Daily␣return")

df_filter <- df[!(df$Coin == "Dogecoin" & df$Date == "2021-01-28"), ]
df_max <- df_filter %>%
  group_by(Coin) %>%
  top_n(1, Daily_Return) %>%
  ungroup()
print(df_max)

# Plot boxplot for each type
ggplot(df_filter , aes(x = Coin, y = Daily_Return)) +
  geom_boxplot() +
  geom_text(data = df_max, aes(label = round(Daily_Return, 2)),
            position = position_dodge(width = 0.5),vjust = -0.5, color = "red") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black")) +
  labs(x = "Cryptocurrency", y = "Daily␣return")

# Reorder Coin factor to control facet order
df_filter$Coin <- factor(df_filter$Coin, levels = c("Bitcoin", "Ethereum",
"EthereumClassic", "XRP", "Cardano", "Dogecoin","TetherUSDt"))
unique(df_filter$Coin)

# Plot with 3 columns (first row will have 3, second will wrap with 4)
ggplot(df_filter, aes(x = Daily_Return, fill = Coin)) +
  geom_histogram(binwidth = 0.01, alpha = 0.5) +
  scale_x_continuous(limits = c(-0.5, 0.5)) +
  labs(x = "Daily␣Return", y = "Frequency") +
```

```
  facet_wrap(~ Coin, nrow = 3, scales = "free_y")

ggplot(df_filter, aes(x = Daily_Return, fill = Coin)) +
  geom_histogram(binwidth = 0.01, alpha = 0.5, color = "black") +
  scale_x_continuous(limits = c(-0.5, 0.5)) +
  scale_fill_grey(start = 0.2, end = 0.8) +
  labs(x = "Daily␣Return", y = "Frequency") +
  facet_wrap(~ Coin, nrow = 3, scales = "free_y") +
  theme_minimal() +
  theme(legend.position = "none")

# Updates of Plot histogram for the specific Coin
CoinName <- 'Bitcoin'
CoinName <- 'Ethereum'
CoinName <- 'EthereumClassic'
CoinName <- 'Cardano'
CoinName <- 'TetherUSDt'
CoinName <- 'XRP'
CoinName <- 'Dogecoin'

#Distribution of Daily Return: CoinName
ggplot(df[df$Coin == CoinName, ], aes(x = Daily_Return)) +
  geom_histogram(binwidth = 0.001, alpha = 0.5) +
  scale_x_continuous(breaks = seq(-1, 5, 0.1), limits = c(-0.5, 0.5)) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
  labs( x = "Daily␣Returns", y = 'Frequency') +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"))

#Weekly Moving Average of Prices: CoinName
ggplot(df[df$Coin == CoinName, ], aes(x = Date, y = MAvgClose )) +
  geom_line() +
  labs(x = "Date", y = "Closing␣Price") +
  theme(plot.title = element_text(hjust = 0.5),panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
```

```
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"))

#Weekly Moving Average of Daily Returns:CoinName
ggplot(df[df$Coin == CoinName, ], aes(x = Date, y = MAvgReturn)) +
  geom_line() +
  geom_hline(yintercept = 0, color = "red") +
  labs( x = "Date", y = "Daily␣return") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black")) +
  ylim(-0.1, 0.25)

#Line chart of the moving average standardized prices by date
# Updated custom mapping
line_types <- c("Bitcoin" = "solid",
                "Cardano" = "dashed",
                "Dogecoin" = "solid", # changed from dotdash to solid
                "Ethereum" = "dotted",
                "EthereumClassic" = "longdash",
                "TetherUSDt" = "twodash",
                "XRP" = "dotted")

line_sizes <- c("Bitcoin" = 0.5,
                "Cardano" = 0.8,
                "Dogecoin" = 1, # slightly thicker to distinguish
                "Ethereum" = 0.8,
                "EthereumClassic" = 0.5,
                "TetherUSDt" = 0.5,
                "XRP" = 0.5)

ggplot(df, aes(x = Date, y = MAvgStdPrice, group = Coin)) +
  geom_line(aes(linetype = Coin, size = Coin), color = "black") +
  labs(x = "Date", y = "Standardized␣Price") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(hjust = 0.5),
```

```
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    axis.line.x = element_line(color = "black"), # Add x-axis line
    axis.line.y = element_line(color = "black")
  ) +
  scale_linetype_manual(name = "cryptocurrency", values = line_types) +
  scale_size_manual(name = "cryptocurrency", values = line_sizes) +
  scale_x_date(
    date_labels = "%Y-%m",
    breaks = "1␣year",
    limits = as.Date(c("2017-11-01", "2024-06-30"))
  )

#Line chart of the moving average standardized prices by date excluding Dogecoin
ggplot(subset(df, Coin != "Dogecoin"), aes(x = Date, y = MAvgStdPrice, group = Coin)) +
  geom_line(aes(linetype = Coin, size = Coin), color = "black") +
  labs(x = "Date", y = "Standardized␣Price") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(hjust = 0.5),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    axis.line.x = element_line(color = "black"), # Add x-axis line
    axis.line.y = element_line(color = "black")
  ) +
  scale_linetype_manual(name = "cryptocurrency", values = line_types) +
  scale_size_manual(name = "cryptocurrency", values = line_sizes) +
  scale_x_date(
    date_labels = "%Y-%m",
    breaks = "1␣year",
    limits = as.Date(c("2017-11-01", "2024-06-30"))
  )

price2023 <- subset(df, Date == '2022-12-31', select = c("Coin", "Close"))
names(price2023)[names(price2023) == "Close"] <- "daytPrice"
df <- merge(df,price2023, by = 'Coin')
```

```
df$StdPrice2023 <- df$Close*100/df$daytPrice

#Line chart of the standardized prices by date, using 2022-12-31 as day0
ggplot(df[df$Date >= '2022-12-31',], aes(x = Date, y = StdPrice2023, group = Coin)) +
  geom_line(aes(linetype = Coin, size = Coin), color = "black") +
  labs(x = "Date", y = "Standardized Price") +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(hjust = 0.5),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    axis.line.x = element_line(color = "black"), # Add x-axis line
    axis.line.y = element_line(color = "black")
  ) +
  scale_linetype_manual(name = "cryptocurrency", values = line_types) +
  scale_size_manual(name = "cryptocurrency", values = line_sizes) +
  scale_x_date(
    date_labels = "%Y-%m",
    breaks = "3 months",
    limits = as.Date(c("2022-11-01", "2024-06-30"))
  )

# Reshape the dataframe
df_DailyReturn <- pivot_wider(df[complete.cases(df), c("Date", "Coin",
"Daily_Return")], names_from = Coin, values_from = Daily_Return)
head(df_DailyReturn)
```

A.2 Corelation and Association

```
# Compute correlation matrix
cor_pearson <- cor(df_DailyReturn[, !(names(df_DailyReturn) == "Date")])
cor_pearson
corrplot(cor_pearson,method = 'shade',
        addCoef.col = 'black', tl.col = "black", mar = c(0,0,0,0),
        addgrid.col = 'white',order = "FPC")
```

```
cor_kendall <- cor(df_DailyReturn[, !(names(df_DailyReturn) == "Date")]
,method = 'kendall')
cor_kendall
corrplot(cor_kendall,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white', order = "FPC")

cor_spearman <- cor(df_DailyReturn[, !(names(df_DailyReturn) == "Date")],
method = 'spearman')
cor_spearman
corrplot(cor_spearman,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white', order = "FPC")

#Compute the cross rank increment correlation coefficient xi
cor_xi <- xicor(df_DailyReturn[, !(names(df_DailyReturn) == "Date")], ties = TRUE)
cor_xi
corxi2<-cor_xi/cor_xi[1,1]
corxi2[7,7] <-1
corxi2

desired_order <- c("Ethereum", "Bitcoin", "Cardano", "EthereumClassic", "XRP",
"Dogecoin", "TetherUSDt")
# Reorder the correlation matrix
cor_xi_ordered <- cor_xi[desired_order, desired_order]
corrplot(cor_xi_ordered,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white')

# Compute the xicor and p-values for each test
results <- data.frame(xi = NA, sd = NA, pval = NA)
results[1, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$Ethereum,
pvalue = TRUE, ties = TRUE)
results[2, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$EthereumClassic,
pvalue = TRUE, ties = TRUE)
results[3, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$TetherUSDt,
pvalue = TRUE, ties = TRUE)
results[4, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$XRP,
pvalue = TRUE, ties = TRUE)
results[5, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$Cardano,
pvalue = TRUE, ties = TRUE)
results[6, ] <- xicor(df_DailyReturn$Bitcoin, df_DailyReturn$Dogecoin,
pvalue = TRUE, ties = TRUE)
```

```
# Create a table with the results
table <- data.frame(
  Cryptocurrency1 = c("Bitcoin", "Bitcoin", "Bitcoin", "Bitcoin", "Bitcoin", "Bitcoin"),
  Cryptocurrency2 = c("Ethereum", "EthereumClassic", "TetherUSDt", "XRP", "Cardano",
  "Dogecoin"),
  xi = results[, 1],
  pval = results[, 3]
)
print(table)

# Reshape the dataframe
df_StdPrice <- pivot_wider(df[complete.cases(df), c("Date", "Coin", "StdPrice")],
names_from = Coin, values_from = StdPrice)
head(df_StdPrice)

# Compute correlation matrix
cor_pearson <- cor(df_StdPrice[, !(names(df_StdPrice) == "Date")])
corrplot(cor_pearson,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white', order = "FPC")

cor_kendall <- cor(df_StdPrice[, !(names(df_StdPrice) == "Date")],method = 'kendall')
cor_kendall
corrplot(cor_kendall,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white', order = "FPC")

cor_spearman <- cor(df_StdPrice[, !(names(df_StdPrice) == "Date")],method = 'spearman')
cor_spearman
corrplot(cor_spearman,method = 'shade',addCoef.col = 'black', tl.col = "black",
mar = c(0,0,0,0),addgrid.col = 'white', order = "FPC")

#Compute the cross rank increment correlation coefficient xi
cor_xi <- xicor(df_StdPrice[, !(names(df_StdPrice) == "Date")], ties = TRUE)
cor_xi

# Reorder the correlation matrix
desired_order <- c("Ethereum", "Dogecoin", "Cardano","Bitcoin", "EthereumClassic",
"XRP", "TetherUSDt")
cor_xi_ordered <- cor_xi[desired_order, desired_order]
corrplot(cor_xi_ordered,method = 'shade',addCoef.col = 'black', tl.col = "black",
```

```
mar = c(0,0,0,0),addgrid.col = 'white')
mtext("Chatterjee␣Correlation␣Matrix", at=3.5, line=0, cex=1.2)
```

A.3 Tests for Normality

```
# Define the list of coin names
coinNames <- c("Bitcoin", "Ethereum", "EthereumClassic", "Cardano", "TetherUSDt",
"XRP", "Dogecoin")

# Create an empty data frame to store the results
results <- data.frame(CoinName = character(),
    ShapiroWilkPValue = numeric(),
    KolmogorovSmirnovPValue = numeric(),
    QQPlot = character(),
    stringsAsFactors = FALSE)

# Set up multiple plot windows
par(mfrow = c(1, 1))

# Iterate over each coin name and perform the tests
for (coinName in coinNames) {
  # Perform Shapiro-Wilk test
  shapiroResult <- shapiro.test(df[df$Coin == coinName, "Daily_Return"])
  # Remove ties from the data
  uniqueData <- unique(df[df$Coin == coinName, "Daily_Return"])
  # Perform Kolmogorov-Smirnov test
  ksResult <- ks.test(uniqueData, 'pnorm')
  # Create Q-Q plot
  qqPlot <- qqnorm(df[df$Coin == coinName, "Daily_Return"],
 main = NULL,       # Remove title
 ylab = paste(coinName, "Sample␣Quantiles"),
 pch = 20)          # Use dots instead of circles
  # Add Q-Q line
  qqline(df[df$Coin == coinName, "Daily_Return"])
  # Append the results to the data frame
  results <- rbind(results, data.frame(CoinName = coinName,
   ShapiroWilkPValue = shapiroResult$p.value,
   KolmogorovSmirnovPValue = ksResult$p.value
```

```
  ),
  stringsAsFactors = FALSE)
}
# Print the results table
print(results)

for (coinName in coinNames) {
  # Perform Shapiro-Wilk test
  shapiroResult <- shapiro.test(df[df$Coin == coinName, "Daily_Return"])
  print(coinName)
  print(shapiroResult)
  # Remove ties from the data
  uniqueData <- unique(df[df$Coin == coinName, "Daily_Return"])
  # Perform Kolmogorov-Smirnov test
  ksResult <- ks.test(uniqueData, 'pnorm')
  print(ksResult)
}
# Perform the multivariate normality test
library(MVN)
MVNDataset <- df_DailyReturn[, !(names(df_DailyReturn) %in% c("TetherUSDt", "Date"))]
head(MVNDataset)
MVNTest <- mvn(MVNDataset,
               mvnTest = "mardia",
               multivariatePlot = "qq")

#### Multivariate Normality Result
MVNTest$multivariateNormality
max_rows <- which(MVNDataset$Dogecoin == max(MVNDataset$Dogecoin))
MVNDataset[max_rows, ]
df_cleaned <- MVNDataset[-max_rows, ]
MVNTest <- mvn(df_cleaned,
               mvnTest = "mardia",
               multivariatePlot = "qq")
#### Multivariate Normality Result
MVNTest$multivariateNormality
print(MVNTest)
```

APPENDIX B

SOURCE CODE OF CHAPTER THREE

```
library(corrr)
library(ggcorrplot)
library(FactoMineR)
library(factoextra)

DailyReturn <- df_DailyReturn[, !(names(df_DailyReturn) == "Date")]
DailyReturn <- DailyReturn[, !(names(DailyReturn) == "TetherUSDt")]
head(DailyReturn)

#standardize the data
data_normalized <- scale(DailyReturn)
head(data_normalized)

#correlation and covariance matrix
cor(DailyReturn)
cor(data_normalized)
cov(DailyReturn)
cov(data_normalized)

#Apply PCA
data.pca <- princomp(DailyReturn) #PCA on covariance matrix and raw data
data.pca <- princomp(data_normalized) #PCA on covariance matrix
data.pca <- princomp(DailyReturn, cor = TRUE) #PCA on correlation matrix
data.pca <- princomp(data_normalized, cor = TRUE)
summary(data.pca)

#Scree Plot to visualize the importance of each principal component
fviz_eig(data.pca, addlabels = TRUE,main = "")
#exploring the loadings of each principal component
data.pca$loadings[, 1:3]
#Biplot to visualize the similarities and dissimilarities between the samples
fviz_pca_var(data.pca, col.var = "black")
#determine how much each variable is represented in a given component
fviz_cos2(data.pca, choice = "var", axes = 1:3)

p <- fviz_cos2(data.pca, choice = "var", axes = 1:3)
p + ggtitle(NULL)
p <- fviz_cos2(data.pca, choice = "var", axes = 1:2)
```

```
p + ggtitle(NULL)

eigvals <- data.pca$sdev^2
L <- unclass(data.pca$loadings)
coords <- L %*% diag(data.pca$sdev)
coords

#Biplot combined with cos2
fviz_pca_var(data.pca, col.var = "cos2",
            gradient.cols = c("black", "orange", "green"),
            repel = TRUE)
p <- fviz_pca_var(data.pca,
col.var = "cos2",
gradient.cols = c("black", "orange", "green"),
repel = TRUE)
p + ggtitle(NULL) # Removes the title

# Extract Cos2 values
var <- get_pca_var(data.pca)
cos2 <- var$cos2
# Get the original variances of the variables
variances <- rowSums(cos2)
# Compute the sum of Cos2 across the first 3 PCs
scos2 <- rowSums(cos2[, 1:3])
scos2 <- rowSums(cos2[, 1:2])

# Standardize the SCos2 values
standardized_scos2 <- scos2 / variances
standardized_scos2
# Create a bar plot
barplot(
  sort(standardized_scos2, decreasing = TRUE),
  horiz = FALSE,
  col = "steelblue",
  las = 2,
  ylab = "Standardized␣SCos2",
  main = ""
)
dfcos2 <- data.frame(
  Variable = names(standardized_scos2),
```

```
  SCos2_Standardized = standardized_scos2
)
# Sort by value for better visual order
dfcos2 <- dfcos2[order(dfcos2$SCos2_Standardized, decreasing = TRUE), ]

# Create bar plot
ggplot(dfcos2, aes(x = reorder(Variable, -SCos2_Standardized),
y = SCos2_Standardized)) +
  geom_bar(stat = "identity", fill = "steelblue",width = 0.7) +
  labs(
    title = "",
    x = NULL,
    y = "Standardized␣SCos2"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# PCA on standardized prices
df_StdPrice <- pivot_wider(df[, c("Date", "Coin", "StdPrice")], names_from = Coin,
values_from = StdPrice)
df_StdPrice <- df_StdPrice[, !(names(df_StdPrice) %in% c("TetherUSDt", "Date"))]
head(df_StdPrice)

#correlation and covariance matrix
cor(df_StdPrice)
cov(df_StdPrice)

#Apply PCA
data.pca <- princomp(df_StdPrice)
data.pca <- princomp(df_StdPrice, cor = TRUE)
summary(data.pca)

#Scree Plot to visualize the importance of each principal component
fviz_eig(data.pca, addlabels = TRUE, main = "")
#exploring the loadings of each principal component
data.pca$loadings[, 1:3]
#Biplot to visualize the similarities and dissimilarities between the samples
fviz_pca_var(data.pca, col.var = "black")

#determine how much each variable is represented in a given component
```

```
fviz_cos2(data.pca, choice = "var", axes = 1:2)
p <- fviz_cos2(data.pca, choice = "var", axes = 1:2)
p + ggtitle(NULL)
p <- fviz_cos2(data.pca, choice = "var", axes = 1:3)
p + ggtitle(NULL)

#Biplot combined with cos2
fviz_pca_var(data.pca, col.var = "cos2",
            gradient.cols = c("black", "orange", "green"),
            repel = TRUE)
p <- fviz_pca_var(data.pca,
col.var = "cos2",
gradient.cols = c("black", "orange", "green"),
repel = TRUE)
p + ggtitle(NULL) # Removes the title

# Extract Cos2 values
var <- get_pca_var(data.pca)
cos2 <- var$cos2

# Get the original variances of the variables
#variances <- apply(DailyReturn, 2, var)
variances <- rowSums(cos2)

# Compute the sum of Cos2 across the first k PCs
scos2 <- rowSums(cos2[, 1:3])
scos2 <- rowSums(cos2[, 1:2])

# Standardize the SCos2 values
standardized_scos2 <- scos2 / variances
standardized_scos2
# Create a bar plot
barplot(
  sort(standardized_scos2, decreasing = TRUE),
  horiz = FALSE,
  col = "steelblue",
  las = 2,
  ylab = "Standardized␣SCos2",
  main = ""
)
```

```
dfcos2 <- data.frame(
  Variable = names(standardized_scos2),
  SCos2_Standardized = standardized_scos2
)

# Sort by value for better visual order
dfcos2 <- dfcos2[order(dfcos2$SCos2_Standardized, decreasing = TRUE), ]

# Create bar plot
ggplot(dfcos2, aes(x = reorder(Variable, -SCos2_Standardized),
y = SCos2_Standardized)) +
  geom_bar(stat = "identity", fill = "steelblue",width = 0.7) +
  labs(
    title = "",
    x = NULL,
    y = "Standardized␣SCos2"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

B.2 Factor Analysis

```
factor_return <- factanal(DailyReturn, factors = 2,rotation = "varimax")
factor_return <- factanal(DailyReturn, factors = 3,rotation = "varimax")
factor_return
for (i in 1:6) {
  result <- factanal(DailyReturn, factors = i,rotation = "varimax")
  cat("Factors:", i, "Degrees␣of␣freedom",result$dof,"␣-␣p-value:",
  result$PVAL, "\n")
}

# Factor analysis on Standardized prices data
factor_return <- factanal(df_StdPrice, factors = 2,rotation = "varimax")
factor_return <- factanal(df_StdPrice, factors = 3,rotation = "varimax")
factor_return

# Factor analysis using principal component method
library(psych)
# Perform factor analysis with principal components
```

```
fa_return <- fa(DailyReturn, nfactors = 2, fm = "pc", rotate = "varimax")
fa_return <- fa(DailyReturn, nfactors = 2, fm = "ml", rotate = "varimax")
fa_return
fa.diagram(fa_return)
```

B.3 Dynamic Factor Analysis

```
####### Dynamic factor analysis ########
library(dfms)
library(MARSS)
library(ggplot2)
library(dplyr)
library(tidyr)
library(scales)
library(dfms)

return_df <- return_df %>% arrange(Date)
tickers <- setdiff(names(return_df), "Date")
ret_xts <- xts(return_df[, tickers], order.by = return_df$Date)

# drop stablecoin (low variance can dominate scaling)
ret_xts <- ret_xts[, setdiff(colnames(ret_xts), "TetherUSDtClose_return")]

# ensure numeric and finite
ret_xts <- ret_xts %>% as.matrix() %>% apply(2, function(x) as.numeric(x))
ret_xts <- xts(ret_xts, order.by = return_df$Date)

# standardize (dfms typically assumes standardized series)
X <- scale(coredata(ret_xts)) # rows=time, cols=assets
dates <- index(ret_xts)

#Optimal Number of Factors (r)
ic <- ICr(X)
print(ic)
plot(ic)
screeplot(ic)

# Using VARselect() with 2 principal components to estimate the VAR lag order
vars::VARselect(ic$F_pca[, 1:2])
```

```
# Fit DFM with r factors and p lags in the transition equation
fit_dfm <- DFM(
  X              = X,     # T x N standardized data
  r              = 2,     # factors
  p              = 3,     # factor ar lags
  q              = 1,     # idiosyncratic ar lags
  standardize    = FALSE, # already standardized above
  em.iters       = 1000,
  conv.tol       = 1e-6,
  verbose        = TRUE
)
summary(fit_dfm)
plot(fit_dfm)
```

APPENDIX C

SOURCE CODE OF CHAPTER FOUR

```
# Import finance index datasets, and Coins datasets
# Read the CSV files
EUR_USD <- read.csv("EUR_USD␣Historical␣Data.csv", header = TRUE)
USD_JPY <- read.csv("USD_JPY␣Historical␣Data.csv", header = TRUE)
GBP_USD <- read.csv("GBP_USD␣Historical␣Data.csv", header = TRUE)
CHF_USD <- read.csv("CHF_USD␣Historical␣Data.csv", header = TRUE)
CAD_USD <- read.csv("CAD_USD␣Historical␣Data.csv", header = TRUE)
AUD_USD <- read.csv("AUD_USD␣Historical␣Data.csv", header = TRUE)
NZD_USD <- read.csv("NZD_USD␣Historical␣Data.csv", header = TRUE)
EUR_GBP <- read.csv("EUR_GBP␣Historical␣Data.csv", header = TRUE)
EUR_JPY <- read.csv("EUR_JPY␣Historical␣Data.csv", header = TRUE)
GBP_JPY <- read.csv("GBP_JPY␣Historical␣Data.csv", header = TRUE)
AUD_JPY <- read.csv("AUD_JPY␣Historical␣Data.csv", header = TRUE)
CHF_JPY <- read.csv("CHF_JPY␣Historical␣Data.csv", header = TRUE)
SP500 <- read.csv("S&P␣500␣Historical␣Data.csv", header = TRUE)
DowJones <- read.csv("Dow␣Jones␣Industrial␣Average␣Historical␣Data.csv",
header = TRUE)
NASDAQ <- read.csv("NASDAQ␣Composite␣Historical␣Data.csv", header = TRUE)
Russell2000 <- read.csv("Russell␣2000␣Historical␣Data.csv", header = TRUE)
XAU_USD <- read.csv("XAU_USD␣Historical␣Data.csv", header = TRUE)

# Read the Text files
Nasdaq100 <- read.table("Nasdaq␣100␣Index.txt", header = FALSE, sep = "␣" )
DJOilGas <- read.table("DJ␣US␣Oil␣Gas␣Index.txt", header = FALSE, sep = "␣" )
DJGoldMining <- read.table("DJ␣US␣Gold␣Mining␣Index.txt", header = FALSE
, sep = "␣" )

# Read coin file
Bitcoin <- read.csv("BTC-USD.csv", header = TRUE)
Ethereum <- read.csv("ETH-USD.csv", header = TRUE)
EthereumClassic <- read.csv("ETC-USD.csv", header = TRUE)
TetherUSDt <- read.csv("USDT-USD.csv", header = TRUE)
XRP <- read.csv("XRP-USD.csv", header = TRUE)
Cardano <- read.csv("ADA-USD.csv", header = TRUE)
Dogecoin <- read.csv("DOGE-USD.csv", header = TRUE)

# Data Cleaning and Pre-processing
# Assign the column names to the first eight main columns of Text files
```

```
column_names <- c("Symbol", "Date","Open","Price","High","Low","Volume","Company")
names(DJOilGas)[1:8] <- column_names
names(Nasdaq100)[1:8] <- column_names
names(DJGoldMining)[1:8] <- column_names

# Rename the column names to the csv files
column_names <- c("Date","Price","Open","High","Low","Volume","Change")
names(EUR_USD) <- column_names
names(USD_JPY) <- column_names
names(GBP_USD) <- column_names
names(CHF_USD) <- column_names
names(CAD_USD) <- column_names
names(AUD_USD) <- column_names
names(NZD_USD) <- column_names
names(EUR_GBP) <- column_names
names(EUR_JPY) <- column_names
names(GBP_JPY) <- column_names
names(AUD_JPY) <- column_names
names(CHF_JPY) <- column_names
names(SP500) <- column_names
names(DowJones) <- column_names
names(NASDAQ) <- column_names
names(XAU_USD) <- column_names
names(Russell2000) <- c("Date","Price","Open","High","Low")

# Define a Function to Convert the Date Column
convert_date_column <- function(df, date_column_name = "Date",
date_format = "%m/%d/%Y")
{  df[[date_column_name]] <- as.Date(df[[date_column_name]], format = date_format)
  return(df) }

# List of data frame names
data_frames <- c("EUR_USD", "USD_JPY", "GBP_USD", "CHF_USD", "CAD_USD",
                 "AUD_USD", "NZD_USD", "EUR_GBP", "EUR_JPY", "GBP_JPY",
                 "AUD_JPY", "CHF_JPY", "SP500", "DowJones", "NASDAQ",
                 "Russell2000", "XAU_USD", "Nasdaq100", "DJOilGas", "DJGoldMining")

# Loop through each data frame and convert the Date column
for (df_name in data_frames) {
  df <- get(df_name)
```

```
  df <- convert_date_column(df)
  assign(df_name, df) }

# View the first few rows of each updated data frame
for (df_name in data_frames) {
  cat(paste("Head␣of", df_name, ":\n"))
  print(head(get(df_name)))
  cat("\n") }

# Function to rename the 'Price' column and select only 'Date' and new 'Close' column
rename_and_select_columns <- function(df, df_name) {
  new_price_name <- paste0(df_name, "_close")
  df %>%
    rename_at(vars(Price), ~ new_price_name) %>%
    select(Date, new_price_name) }

# Initialize a list to hold the renamed data frames
renamed_dfs <- list()

# Loop through each data frame to rename 'Price' column and add to list
for (df_name in data_frames) {
  df <- get(df_name)
  df <- rename_and_select_columns(df, df_name)
  renamed_dfs[[df_name]] <- df }

# Merge all data frames by 'Date'
financeIndex <- Reduce(function(x, y) full_join(x, y, by = "Date"), renamed_dfs)

# Convert string to numeric after removing commas
financeIndex$SP500_close <- as.numeric(gsub(",", "", financeIndex$SP500_close))
financeIndex$DowJones_close <- as.numeric(gsub(",", "", financeIndex$DowJones_close))
financeIndex$NASDAQ_close <- as.numeric(gsub(",", "", financeIndex$NASDAQ_close))
financeIndex$XAU_USD_close <- as.numeric(gsub(",", "", financeIndex$XAU_USD_close))
head(financeIndex)

# Process Bitcoin data
# Convert the Date column to actual date format
Bitcoin$Date <- as.Date(Bitcoin$Date, format = "%Y-%m-%d")
# Rename the column 'Close' to 'CoinNameClose'
Bitcoin <- Bitcoin %>% rename(BitcoinClose = Close)
```

```
# Select only the Date and BitcoinClose columns
Bitcoin_price <- Bitcoin %>% select(Date, BitcoinClose)

# Process Ethereum data
Ethereum$Date <- as.Date(Ethereum$Date, format = "%Y-%m-%d")
Ethereum <- Ethereum %>% rename(EthereumClose = Close)
Ethereum_price <- Ethereum %>% select(Date, EthereumClose)

# Process Ethereum Classic data
EthereumClassic$Date <- as.Date(EthereumClassic$Date, format = "%Y-%m-%d")
EthereumClassic <- EthereumClassic %>% rename(EthereumClassicClose = Close)
EthereumClassic_price <- EthereumClassic %>% select(Date, EthereumClassicClose)

# Process Tether USDt data
TetherUSDt$Date <- as.Date(TetherUSDt$Date, format = "%Y-%m-%d")
TetherUSDt <- TetherUSDt %>% rename(TetherUSDtClose = Close)
TetherUSDt_price <- TetherUSDt %>% select(Date, TetherUSDtClose)

# Process XRP data
XRP$Date <- as.Date(XRP$Date, format = "%Y-%m-%d")
XRP <- XRP %>% rename(XRPClose = Close)
XRP_price <- XRP %>% select(Date, XRPClose)

# Process Cardano data
Cardano$Date <- as.Date(Cardano$Date, format = "%Y-%m-%d")
Cardano <- Cardano %>% rename(CardanoClose = Close)
Cardano_price <- Cardano %>% select(Date, CardanoClose)

# Process Dogecoin data
Dogecoin$Date <- as.Date(Dogecoin$Date, format = "%Y-%m-%d")
Dogecoin <- Dogecoin %>% rename(DogecoinClose = Close)
Dogecoin_price <- Dogecoin %>% select(Date, DogecoinClose)

# Join all cryptocurrency dataframes to financeIndex by Date using inner join
cryptocurrency_df <- financeIndex %>%
  inner_join(Bitcoin_price, by = "Date") %>%
  inner_join(Ethereum_price, by = "Date") %>%
  inner_join(EthereumClassic_price, by = "Date") %>%
  inner_join(TetherUSDt_price, by = "Date") %>%
  inner_join(XRP_price, by = "Date") %>%
```

```
  inner_join(Cardano_price, by = "Date") %>%
  inner_join(Dogecoin_price, by = "Date")
head(cryptocurrency_df)

# Order the data by date
cryptocurrency_df <- cryptocurrency_df %>%arrange(Date)

# List of columns to compute daily returns for
columns_to_compute <- c("EUR_USD_close", "USD_JPY_close", "GBP_USD_close",
"CHF_USD_close", "CAD_USD_close", "AUD_USD_close", "NZD_USD_close",
"EUR_GBP_close", "EUR_JPY_close","GBP_JPY_close", "AUD_JPY_close",
"CHF_JPY_close", "SP500_close", "DowJones_close", "NASDAQ_close",
"Russell2000_close","XAU_USD_close", "Nasdaq100_close", "DJOilGas_close",
"DJGoldMining_close", "BitcoinClose","EthereumClose", "EthereumClassicClose",
"TetherUSDtClose", "XRPClose", "CardanoClose", "DogecoinClose")

# Compute daily returns
cryptocurrency_df <- cryptocurrency_df %>%
  mutate(across(all_of(columns_to_compute),
                ~ (.-lag(.))/lag(.) * 100,
                .names = "{.col}_return"))

# Remove rows with any NA values
cryptocurrency_df <- cryptocurrency_df %>% na.omit()
head(cryptocurrency_df)

# Viewing and Exploring Data
# Extract year and count rows by year
count_by_year <- cryptocurrency_df %>%
  mutate(
    year = format(Date, "%Y"),
    month = format(Date, "%m")
  ) %>%
  group_by(year) %>%
  summarize(count = n(), .groups = 'drop')
print(count_by_year)

# Extract month and count rows by month
count_by_month <- cryptocurrency_df %>%
  mutate(
```

```
    year = format(Date, "%Y"),
    month = format(Date, "%m")
  ) %>%
  group_by( month) %>%
  summarize(count = n(), .groups = 'drop')
print(count_by_month)

# Extract weekday and count rows by weekday
count_by_weekday <- cryptocurrency_df %>%
  mutate(weekday = weekdays(Date)) %>%
  group_by(weekday) %>%
  summarize(count = n(), .groups = 'drop')
print(count_by_weekday)

# Extract year and month, and count rows by year and month
count_by_year_month <- cryptocurrency_df %>%
  mutate(
    year = format(Date, "%Y"),
    month = format(Date, "%m")
  ) %>%
  group_by(year, month) %>%
  summarize(count = n(), .groups = 'drop')

# Defining Shrinkage and Positive Shrinkage estimation functions
Shrinkage_Est <- function(beta_FM,beta_SM,test_stat,p2){
  return (beta_FM - ((beta_FM - beta_SM)*(p2-2)/test_stat))
}
PShrinkage_Est <-function(beta_FM,beta_SM,test_stat ,p2){
  return (beta_SM + max(0,(1-(p2-2)/test_stat))*(beta_FM - beta_SM))
}
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}

## Prediction model on Present Data ####
cryptocurrency_model <- cryptocurrency_df %>%
  select(
    EUR_USD_close_return,
    USD_JPY_close_return,
    GBP_USD_close_return,
    CHF_USD_close_return,
    CAD_USD_close_return,
```

```
    AUD_USD_close_return,
    NZD_USD_close_return,
    EUR_GBP_close_return,
    EUR_JPY_close_return,
    GBP_JPY_close_return,
    AUD_JPY_close_return,
    CHF_JPY_close_return,
    SP500_close_return,
    DowJones_close_return,
    NASDAQ_close_return,
    Russell2000_close_return,
    XAU_USD_close_return,
    Nasdaq100_close_return,
    DJOilGas_close_return,
    DJGoldMining_close_return,
    BitcoinClose_return,
    EthereumClose_return,
    EthereumClassicClose_return,
    TetherUSDtClose_return,
    XRPClose_return,
    CardanoClose_return,
    DogecoinClose_return
  )
head(cryptocurrency_model)

# Assign each coin's return to Response_Coin, ONLY RUN ONE ROE for each coin model
Response_Coin = "BitcoinClose_return"
Response_Coin = "EthereumClose_return"
Response_Coin = "EthereumClassicClose_return"
Response_Coin = "TetherUSDtClose_return"
Response_Coin = "XRPClose_return"
Response_Coin = "CardanoClose_return"
Response_Coin = "DogecoinClose_return"

y <- cryptocurrency_model %>% dplyr :: select(Response_Coin)%>%as.matrix()
X <- cryptocurrency_model %>% dplyr :: select(-Response_Coin)%>%as.matrix()
X_intercept <-cbind(1,X)
raw_data <- data.frame(y,X)
p <- ncol(X)+1
head(raw_data)
```

```
# Fit Full model
full_names <- colnames(X)
xcount.FM <- c(full_names)
y_name <- colnames(y)
Formula_FM <- as.formula(paste(y_name, "␣~␣",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,raw_data)
summary(model_full)
vif(model_full)

#Submodel by BIC
step_model <- step(model_full, direction = "both",k = log(nrow(raw_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
full_names <- colnames(X)
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
Formula_SM.BIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM, collapse = "+")))
summary(step_model)

#Submodel by AIC
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.AIC,
collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y , alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
```

```
p2.lasso <- p-p1.lasso
xcount.SM.lasso <- c(sub_names.lasso)
Formula_SM.lasso <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.lasso,
collapse = "+")))
submodel_lasso <-lm(Formula_SM.lasso,raw_data)
summary(submodel_lasso)
# If LASSO choose no variable, use the below one
Formula_SM.lasso<- as.formula(paste(y_name, "␣~␣1"))

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)

# Set the seed for reproducibility
set.seed(1234)
for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(raw_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- raw_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(y_name) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-y_name) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(y_name) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-y_name) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
```

```
df_train <- data.frame( y_train, X_train)
#The full model fit based on train data
fit_FM <-lm(Formula_FM,df_train)
beta.FM <- fit_FM$coef
beta.FM[is.na(beta.FM)] <- 0
#The sub model fit based on train data
fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
beta.SM.BIC <- rep(0,p)
beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
beta.SM.AIC <- rep(0,p)
beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
beta.SM.lasso <- rep(0,p)
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE, standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
```

```
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge
yhat_df <- data.frame(y_test_scale,yhat_beta.FM,yhat_beta.SM.BIC,
    yhat_beta.S.BIC ,
    yhat_beta.PS.BIC ,
```

```
    yhat_beta.SM.AIC ,
    yhat_beta.S.AIC ,
    yhat_beta.PS.AIC ,
    yhat_beta.SM.lasso ,
    yhat_beta.S.lasso ,
    yhat_beta.PS.lasso ,
    yhat_beta.lasso ,
    yhat_beta.enet ,
    yhat_beta.alasso ,
    yhat_beta.scad ,
    yhat_beta.ridge )
  #Calculate prediction errors of estimators
  PE_values <- c(FM=Prediction_Error(y_test_scale, yhat_beta.FM),
                 SM.BIC=Prediction_Error(y_test_scale, yhat_beta.SM.BIC),
                 S.BIC =Prediction_Error(y_test_scale, yhat_beta.S.BIC),
                 PS.BIC=Prediction_Error(y_test_scale, yhat_beta.PS.BIC),
                 SM.AIC=Prediction_Error(y_test_scale, yhat_beta.SM.AIC),
                 S.AIC =Prediction_Error(y_test_scale, yhat_beta.S.AIC),
                 PS.AIC=Prediction_Error(y_test_scale, yhat_beta.PS.AIC),
                 SM.lasso=Prediction_Error(y_test_scale, yhat_beta.SM.lasso),
                 S.lasso =Prediction_Error(y_test_scale, yhat_beta.S.lasso),
                 PS.lasso=Prediction_Error(y_test_scale, yhat_beta.PS.lasso),
                 LASSO = Prediction_Error( y_test_scale, yhat_beta.lasso),
                 ENET = Prediction_Error( y_test_scale, yhat_beta.enet),
                 ALASSO = Prediction_Error( y_test_scale, yhat_beta.alasso),
                 SCAD = Prediction_Error( y_test_scale , yhat_beta.scad ),
                 RIDGE = Prediction_Error( y_test_scale , yhat_beta.ridge)
  )
  PE_values_bootstrap[i,] <- PE_values
}
PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))

# Add row names
row_names <- c("Full␣Model","Submodel␣BIC","Shrinkage␣Estimator␣BIC",
"Positive␣Shrinkage␣Estimator␣BIC","Submodel␣AIC","Shrinkage␣Estimator␣AIC",
"Positive␣Shrinkage␣Estimator␣AIC","Submodel␣LASSO","Shrinkage␣Estimator␣LASSO",
"Positive␣Shrinkage␣Estimator␣LASSO", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
```

```
print_with_title <- function(df) {
  cat("Response␣Cryptocurrency:", Response_Coin, "\n\n")
  print(df) }
print_with_title(PE.result)

### Prediction model for Next one day
head(cryptocurrency_df)
cryptocurrency_nextday_df <- cryptocurrency_df %>%
  select(
    Date,
    EUR_USD_close_return,
    USD_JPY_close_return,
    GBP_USD_close_return,
    CHF_USD_close_return,
    CAD_USD_close_return,
    AUD_USD_close_return,
    NZD_USD_close_return,
    EUR_GBP_close_return,
    EUR_JPY_close_return,
    GBP_JPY_close_return,
    AUD_JPY_close_return,
    CHF_JPY_close_return,
    SP500_close_return,
    DowJones_close_return,
    NASDAQ_close_return,
    Russell2000_close_return,
    XAU_USD_close_return,
    Nasdaq100_close_return,
    DJOilGas_close_return,
    DJGoldMining_close_return,
    BitcoinClose_return,
    EthereumClose_return,
    EthereumClassicClose_return,
    TetherUSDtClose_return,
    XRPClose_return,
    CardanoClose_return,
    DogecoinClose_return
  )
head(cryptocurrency_nextday_df)
str(cryptocurrency_nextday_df)
```

```
# Assign each coin's return to Response_Coin, ONLY RUN ONE ROE for each coin model
Response_Coin = "BitcoinClose_return"
Response_Coin = "EthereumClose_return"
Response_Coin = "EthereumClassicClose_return"
Response_Coin = "TetherUSDtClose_return"
Response_Coin = "XRPClose_return"
Response_Coin = "CardanoClose_return"
Response_Coin = "DogecoinClose_return"

cryptocurrency_nextday_model <- cryptocurrency_nextday_df %>%
  arrange(Date) %>%
  mutate(y_Coin = lead(!!sym(Response_Coin)))
cryptocurrency_nextday_model <-
cryptocurrency_nextday_model[complete.cases(cryptocurrency_nextday_model),]
y <- cryptocurrency_nextday_model %>% dplyr :: select(y_Coin)%>%as.matrix()
X <- cryptocurrency_nextday_model %>% dplyr :: select(-c(y_Coin,Date))%>%as.matrix()
X_intercept <-cbind(1,X)
raw_data <- data.frame(y,X)
p <- ncol(X)+1
head(raw_data)

# Fit Full model
full_names <- colnames(X)
xcount.FM <- c(full_names)
y_name <- colnames(y)
Formula_FM <- as.formula(paste(y_name, "␣~␣",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,raw_data)
summary(model_full)
vif(model_full)

#Submodel by BIC
step_model <- step(model_full, direction = "both",k = log(nrow(raw_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
full_names <- colnames(X)
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
```

```
Formula_SM.BIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM,
collapse = "+")))
summary(step_model)
# If BIC choose no variable, use the below one
Formula_SM.BIC<- as.formula(paste(y_name, "␣~␣1"))

#Submodel by AIC
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.AIC,
collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y , alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
p2.lasso <- p-p1.lasso
xcount.SM.lasso <- c(sub_names.lasso)
Formula_SM.lasso <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.lasso,
collapse = "+")))
submodel_lasso <-lm(Formula_SM.lasso,raw_data)
summary(submodel_lasso)
# If LASSO choose no variable, use the below one
Formula_SM.lasso<- as.formula(paste(y_name, "␣~␣1"))

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)
```

```
# Set the seed for reproducibility
set.seed(1234)
for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(raw_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- raw_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(y_name) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-y_name) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(y_name) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-y_name) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
  df_train <- data.frame( y_train, X_train)
  #The full model fit based on train data
  fit_FM <-lm(Formula_FM,df_train)
  beta.FM <- fit_FM$coef
  beta.FM[is.na(beta.FM)] <- 0
  #The sub model fit based on train data
  fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
  beta.SM.BIC <- rep(0,p)
  beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
  beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
  fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
  beta.SM.AIC <- rep(0,p)
  beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
  beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
  fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
  beta.SM.lasso <- rep(0,p)
```

```
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE,
standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)
#the penalty parameter is tuned to obtain optimal prediction
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
```

```
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso,
test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso,
test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge
yhat_df <- data.frame(y_test_scale,yhat_beta.FM,yhat_beta.SM.BIC,
    yhat_beta.S.BIC ,
    yhat_beta.PS.BIC ,
    yhat_beta.SM.AIC ,
    yhat_beta.S.AIC ,
    yhat_beta.PS.AIC ,
    yhat_beta.SM.lasso ,
    yhat_beta.S.lasso ,
    yhat_beta.PS.lasso ,
    yhat_beta.lasso ,
    yhat_beta.enet ,
    yhat_beta.alasso ,
    yhat_beta.scad ,
    yhat_beta.ridge )
#Calculate prediction errors of estimators
```

```
  PE_values <- c(FM=Prediction_Error(y_test_scale, yhat_beta.FM),
                 SM.BIC=Prediction_Error(y_test_scale, yhat_beta.SM.BIC),
                 S.BIC =Prediction_Error(y_test_scale, yhat_beta.S.BIC),
                 PS.BIC=Prediction_Error(y_test_scale, yhat_beta.PS.BIC),
                 SM.AIC=Prediction_Error(y_test_scale, yhat_beta.SM.AIC),
                 S.AIC =Prediction_Error(y_test_scale, yhat_beta.S.AIC),
                 PS.AIC=Prediction_Error(y_test_scale, yhat_beta.PS.AIC),
                 SM.lasso=Prediction_Error(y_test_scale, yhat_beta.SM.lasso),
                 S.lasso =Prediction_Error(y_test_scale, yhat_beta.S.lasso),
                 PS.lasso=Prediction_Error(y_test_scale, yhat_beta.PS.lasso),
                 LASSO = Prediction_Error( y_test_scale, yhat_beta.lasso),
                 ENET = Prediction_Error( y_test_scale, yhat_beta.enet),
                 ALASSO = Prediction_Error( y_test_scale, yhat_beta.alasso),
                 SCAD = Prediction_Error( y_test_scale , yhat_beta.scad ),
                 RIDGE = Prediction_Error( y_test_scale , yhat_beta.ridge)
  )
  PE_values_bootstrap[i,] <- PE_values
}
PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))

# Add row names
row_names <- c("Full␣Model","Submodel␣BIC","Shrinkage␣Estimator␣BIC",
"Positive␣Shrinkage␣Estimator␣BIC","Submodel␣AIC","Shrinkage␣Estimator␣AIC",
"Positive␣Shrinkage␣Estimator␣AIC","Submodel␣LASSO",
"Shrinkage␣Estimator␣LASSO","Positive␣Shrinkage␣Estimator␣LASSO",
"LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
print_with_title <- function(df) {
  cat("Response␣Cryptocurrency:", Response_Coin, "\n\n")
  print(df) }
print_with_title(PE.result)

# Chatterjee Correlation Matrix for daily returns of finance index and coins
correlation_df <- cryptocurrency_model
cor_xi <- xicor(correlation_df, ties = TRUE)
cor_xi<-cor_xi/cor_xi[1,1]
cor_xi
# Remove '_close_return' from row and column names
```

```
rownames(cor_xi) <- gsub("_close_return|Close_return", "", rownames(cor_xi))
colnames(cor_xi) <- gsub("_close_return|Close_return", "", colnames(cor_xi))
cor_xi

# Calculate the sum of absolute values of each row
row_sums <- apply(abs(cor_xi), 1, sum)
# Order the rows by the calculated measure (from large to small)
ordered_indices <- order(row_sums, decreasing = TRUE)
# Reorder the matrix based on the computed order
cor_xi_ordered <- cor_xi[ordered_indices, ordered_indices]
par(oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0)) # Adjusting outer and inner margins
# Plot the correlation matrix with adjusted size
corrplot(cor_xi_ordered,
        method = 'shade',
        addCoef.col = 'black',
        tl.col = "black",
        tl.cex = 0.5, # Adjust the text label size (larger size)
        number.cex = 0.5, # Adjust the correlation coefficients size
        mar = c(0, 0, 0, 0),
        addgrid.col = 'white')
```

C.2 Prediction Models with Principal Components

```
# Model on Present Data
dailyReturnPresent <- cryptocurrency_df %>%
  select(
    EUR_USD_close_return,
    USD_JPY_close_return,
    GBP_USD_close_return,
    CHF_USD_close_return,
    CAD_USD_close_return,
    AUD_USD_close_return,
    NZD_USD_close_return,
    EUR_GBP_close_return,
    EUR_JPY_close_return,
    GBP_JPY_close_return,
    AUD_JPY_close_return,
    CHF_JPY_close_return,
    SP500_close_return,
```

```
    DowJones_close_return,
    NASDAQ_close_return,
    Russell2000_close_return,
    Nasdaq100_close_return,
    XAU_USD_close_return,
    DJOilGas_close_return,
    DJGoldMining_close_return,
    BitcoinClose_return,
    EthereumClose_return,
    EthereumClassicClose_return,
    TetherUSDtClose_return,
    XRPClose_return,
    CardanoClose_return,
    DogecoinClose_return
  )
head(dailyReturnPresent)

# Perform PCA on currency pairs and extract the first principal component
currency_pairs <- dailyReturnPresent %>%
  select(EUR_USD_close_return, USD_JPY_close_return, GBP_USD_close_return,
  CHF_USD_close_return, CAD_USD_close_return, AUD_USD_close_return,
  NZD_USD_close_return, EUR_GBP_close_return, EUR_JPY_close_return,
  GBP_JPY_close_return, AUD_JPY_close_return, CHF_JPY_close_return)
currency_pca <- princomp(currency_pairs, cor = FALSE) #use covariance matrix
currency_pc1 <- currency_pca$scores[, 1]
summary(currency_pca)

# Perform PCA on stock indices and extract the first principal component
stock_indices <- dailyReturnPresent %>%
  select(SP500_close_return, DowJones_close_return, NASDAQ_close_return,
  Russell2000_close_return, Nasdaq100_close_return)
stock_pca <- princomp(stock_indices, cor = FALSE)
stock_pc1 <- stock_pca$scores[, 1]
summary(stock_pca)

# Perform PCA on commodities and extract the first principal component
goldOil <- dailyReturnPresent %>%
  select(XAU_USD_close_return, DJOilGas_close_return, DJGoldMining_close_return)
goldOil_pca <- princomp(goldOil, cor = FALSE)
goldOils_pc1 <- goldOil_pca$scores[, 1]
```

```
summary(goldOil_pca)

# Combine the first principal components with the cryptocurrency returns
cryptos <- dailyReturnPresent %>%
  select(BitcoinClose_return, EthereumClose_return, EthereumClassicClose_return,
  TetherUSDtClose_return, XRPClose_return, CardanoClose_return, DogecoinClose_return)
dailyReturnPresentPC <- data.frame(
  Currency_PC1 = currency_pc1,
  Stock_PC1 = stock_pc1,
  GoldOils_PC1 = goldOils_pc1,
  cryptos
)
head(dailyReturnPresentPC,5)

# List of response variables
cryptocurrencies <- c("BitcoinClose_return", "EthereumClose_return",
"EthereumClassicClose_return", "TetherUSDtClose_return", "XRPClose_return",
"CardanoClose_return", "DogecoinClose_return")

# Loop through each cryptocurrency and perform linear regression
models <- list()
for (crypto in cryptocurrencies) {
  # Create the formula for the linear regression
  formula <- as.formula(paste(crypto, "~␣."))
  # Perform the linear regression
  model <- lm(formula, data = dailyReturnPresentPC)
  # Store the model in the list
  models[[crypto]] <- model
  # Print summaries of the models
  cat("\nLinear␣regression␣model␣for", crypto, ":\n")
  print(summary(model))
  print(vif(model))
}

# Bootstrap samples and Prediction error
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}
# Number of bootstrap samples
B <- 250
set.seed(1234)
# Store prediction error results
```

```
PE_results <- list()
for (crypto in cryptocurrencies) {
  cat("\nProcessing:", crypto, "\n")
  PE_values_bootstrap <- numeric(B)

  for (i in 1:B) {
    # Bootstrap sample
    bootstrap_indices <- sample(nrow(dailyReturnPresentPC), replace = TRUE)
    bootstrap_sample <- dailyReturnPresentPC[bootstrap_indices, ]
    # Split into train/test
    set.seed(i)
    split <- sample.split(bootstrap_sample[[crypto]], SplitRatio = 0.75)
    train_set <- subset(bootstrap_sample, split == TRUE)
    test_set <- subset(bootstrap_sample, split == FALSE)
    # Fit model on train set
    formula <- as.formula(paste(crypto, "~␣."))
    model <- lm(formula, data = train_set)
    # Predict on test set
    y_test <- test_set[[crypto]]
    yhat <- predict(model, newdata = test_set)
    # Compute prediction error
    PE_values_bootstrap[i] <- Prediction_Error(y_test, yhat)
  }
  # Store average and std dev of prediction error
  PE_avg <- mean(PE_values_bootstrap)
  PE_std <- sd(PE_values_bootstrap)
  print(PE_avg)
  print(PE_std)
  PE_results[[crypto]] <- data.frame(PE_avg = PE_avg, PE_std = PE_std)
}
# Combine results into a single data frame
PE_summary <- do.call(rbind, PE_results)
rownames(PE_summary) <- cryptocurrencies
print(PE_summary)

# Model on Next one day Data
# List of response variables
cryptocurrencies <- c("BitcoinClose_return", "EthereumClose_return",
"EthereumClassicClose_return", "TetherUSDtClose_return", "XRPClose_return",
"CardanoClose_return", "DogecoinClose_return")
```

```
# Create a copy of the dataframe to include lead response variables
lead_df <- dailyReturnPresentPC

# Create lead response variables for each cryptocurrency
for (crypto in cryptocurrencies) {
  lead_df[[paste0(crypto, "_nextDay")]] <- dplyr::lead(lead_df[[crypto]], 1)
}

# Drop the last row with NA in lead response variables
lead_df <- lead_df[!is.na(lead_df[[paste0(cryptocurrencies[1], "_nextDay")]]), ]
head(lead_df)

# Loop through each cryptocurrency and perform linear regression
models <- list()
for (crypto in cryptocurrencies) {
  # Create the formula for the linear regression with lead response variable
  predictors <- c(cryptocurrencies, "Currency_PC1", "Stock_PC1", "GoldOils_PC1")
  formula <- as.formula(paste(paste0(crypto, "_nextDay"), "~",
  paste(predictors, collapse = "␣+␣")))
  # Perform the linear regression
  model <- lm(formula, data = lead_df)
  # Store the model in the list
  models[[crypto]] <- model
  #Print summary of the model
  cat("\nLinear␣regression␣model␣for", paste0(crypto, "_nextDay"), ":\n")
  print(summary(models[[crypto]]))
  print(vif(models[[crypto]]))
}
# Bootstrap samples and Prediction error
# Define predictors
predictors <- c(cryptocurrencies, "Currency_PC1", "Stock_PC1", "GoldOils_PC1")
# Prediction error function
Prediction_Error <- function(y, yhat) {
  mean((y - yhat)^2)
}

# Number of bootstrap samples
B <- 250
set.seed(1234)
```

```
# Store prediction error results
PE_results_nextDay <- list()

for (crypto in cryptocurrencies) {
  response <- paste0(crypto, "_nextDay")
  cat("\nProcessing:", response, "\n")
  PE_values_bootstrap <- numeric(B)

  for (i in 1:B) {
    # Bootstrap sample
    bootstrap_indices <- sample(nrow(lead_df), replace = TRUE)
    bootstrap_sample <- lead_df[bootstrap_indices, ]
    # Split into train/test
    split <- sample.split(bootstrap_sample[[response]], SplitRatio = 0.75)
    train_set <- subset(bootstrap_sample, split == TRUE)
    test_set <- subset(bootstrap_sample, split == FALSE)
    # Fit model on train set
    formula <- as.formula(paste(response, "~", paste(predictors, collapse = "␣+␣")))
    model <- lm(formula, data = train_set)
    # Predict on test set
    y_test <- test_set[[response]]
    yhat <- predict(model, newdata = test_set)
    # Compute prediction error
    PE_values_bootstrap[i] <- Prediction_Error(y_test, yhat)
  }
  # Store average and std dev of prediction error
  PE_avg <- mean(PE_values_bootstrap)
  PE_std <- sd(PE_values_bootstrap)
  print(PE_avg)
  print(PE_std)
  PE_results_nextDay[[response]] <- data.frame(PE_avg = PE_avg, PE_std = PE_std)
}

# Combine results into a single data frame
PE_summary_nextDay <- do.call(rbind, PE_results_nextDay)
rownames(PE_summary_nextDay) <- names(PE_results_nextDay)
print(PE_summary_nextDay)
```

APPENDIX D

SOURCE CODE OF CHAPTER FIVE

```
# Join all cryptocurrency dataframes to financeIndex by Date using inner join
cryptocurrency_df <- financeIndex %>%
  inner_join(Bitcoin_price, by = "Date") %>%
  inner_join(Ethereum_price, by = "Date") %>%
  inner_join(EthereumClassic_price, by = "Date") %>%
  inner_join(TetherUSDt_price, by = "Date") %>%
  inner_join(XRP_price, by = "Date") %>%
  inner_join(Cardano_price, by = "Date") %>%
  inner_join(Dogecoin_price, by = "Date")
crypto_weekly_df <- cryptocurrency_df %>%
  select(
    Date,
    SP500_close,
    DowJones_close,
    NASDAQ_close,
    Russell2000_close,
    Nasdaq100_close,
    DJOilGas_close,
    DJGoldMining_close,
    XAU_USD_close,
    BitcoinClose,
    EthereumClose,
    EthereumClassicClose,
    TetherUSDtClose,
    XRPClose,
    CardanoClose,
    DogecoinClose
  )
# Order the data by date
crypto_weekly_df <- crypto_weekly_df %>%arrange(Date)
# Add a column for the weekday
crypto_weekly_df <- crypto_weekly_df %>%
  mutate(Weekday = wday(Date, label = TRUE))
# Reorder columns to place Weekday beside Date
crypto_weekly_df <- crypto_weekly_df %>%
  select(Date, Weekday, everything())
head(crypto_weekly_df)
```

```
# Function to replace NA values with the previous available value
replace_na_with_previous <- function(x) {
  for (i in 2:length(x)) {
    if (is.na(x[i])) {
      x[i] <- x[i-1]
    }
  }
  return(x)
}
# Replace NA values with previous day's value for numerical columns
crypto_weekly_df <- crypto_weekly_df %>%
  mutate(across(where(is.numeric), replace_na_with_previous))
# Select only the Friday rows
crypto_fridays <- crypto_weekly_df %>%
  filter(Weekday == "Fri")
head(crypto_fridays)

# List of columns to compute returns for
columns_to_compute <- c("SP500_close", "DowJones_close", "NASDAQ_close",
"Russell2000_close", "Nasdaq100_close", "DJOilGas_close", "DJGoldMining_close",
"XAU_USD_close", "BitcoinClose","EthereumClose", "EthereumClassicClose",
"TetherUSDtClose", "XRPClose", "CardanoClose", "DogecoinClose")
# Order the data by date
crypto_fridays <- crypto_fridays %>%arrange(Date)

# (1) Compute weekly returns on Fridays
crypto_fridays_return <- crypto_fridays %>%
  mutate(across(all_of(columns_to_compute),
                ~ (.-lag(.))/lag(.) * 100,
                .names = "{.col}_1WeekReturn"))

# (2) Compute weekly returns based on value from two weeks ago
crypto_fridays_return <- crypto_fridays_return %>%
  mutate(across(all_of(columns_to_compute),
                ~ (.-lag(., 2))/lag(., 2) * 100,
                .names = "{.col}_2WeekReturn"))
dim(crypto_fridays_return)

# Remove rows with any NA values
crypto_fridays_df <- crypto_fridays_return %>% na.omit()
```

```
dim(crypto_fridays_df)
head(crypto_fridays_df)

# Prediction model on Present Data
crypto_fridays_model <- crypto_fridays_df %>%
  select(
    SP500_close_1WeekReturn,
    DowJones_close_1WeekReturn,
    NASDAQ_close_1WeekReturn,
    Russell2000_close_1WeekReturn,
    Nasdaq100_close_1WeekReturn,
    DJOilGas_close_1WeekReturn,
    DJGoldMining_close_1WeekReturn,
    XAU_USD_close_1WeekReturn,
    BitcoinClose_1WeekReturn,
    EthereumClose_1WeekReturn,
    EthereumClassicClose_1WeekReturn,
    TetherUSDtClose_1WeekReturn,
    XRPClose_1WeekReturn,
    CardanoClose_1WeekReturn,
    DogecoinClose_1WeekReturn
  )

# Assign each coin's return to Response_Coin, ONLY RUN ONE ROE for each coin model
Response_Coin = "BitcoinClose_1WeekReturn"
Response_Coin = "EthereumClose_1WeekReturn"
Response_Coin = "EthereumClassicClose_1WeekReturn"
Response_Coin = "TetherUSDtClose_1WeekReturn"
Response_Coin = "XRPClose_1WeekReturn"
Response_Coin = "CardanoClose_1WeekReturn"
Response_Coin = "DogecoinClose_1WeekReturn"

y <- crypto_fridays_model %>% dplyr :: select(Response_Coin)%>%as.matrix()
X <- crypto_fridays_model %>% dplyr :: select(-Response_Coin)%>%as.matrix()
X_intercept <-cbind(1,X)
raw_data <- data.frame(y,X)
p <- ncol(X)+1
head(raw_data)

# Fit Full model
```

```
full_names <- colnames(X)
xcount.FM <- c(full_names)
y_name <- colnames(y)
Formula_FM <- as.formula(paste(y_name, "␣~␣",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,raw_data)
summary(model_full)
vif(model_full)

#Submodel by BIC
step_model <- step(model_full, direction = "both",k = log(nrow(raw_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
full_names <- colnames(X)
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
Formula_SM.BIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM, collapse = "+")))
summary(step_model)

#Submodel by AIC
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.AIC,
collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y , alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
p2.lasso <- p-p1.lasso
xcount.SM.lasso <- c(sub_names.lasso)
```

```
Formula_SM.lasso <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.lasso,
collapse = "+")))
submodel_lasso <-lm(Formula_SM.lasso,raw_data)
summary(submodel_lasso)
# If LASSO choose no variable, use the below one
Formula_SM.lasso<- as.formula(paste(y_name, "␣~␣1"))

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)
# Set the seed for reproducibility
set.seed(1234)
for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(raw_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- raw_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(y_name) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-y_name) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(y_name) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-y_name) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
  df_train <- data.frame( y_train, X_train)
  #The full model fit based on train data
  fit_FM <-lm(Formula_FM,df_train)
```

```
beta.FM <- fit_FM$coef
beta.FM[is.na(beta.FM)] <- 0
#The sub model fit based on train data
fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
beta.SM.BIC <- rep(0,p)
beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
beta.SM.AIC <- rep(0,p)
beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
beta.SM.lasso <- rep(0,p)
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE,
standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
```

```
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge
yhat_df <- data.frame(y_test_scale,yhat_beta.FM,yhat_beta.SM.BIC,
    yhat_beta.S.BIC ,
    yhat_beta.PS.BIC ,
    yhat_beta.SM.AIC ,
    yhat_beta.S.AIC ,
```

```
    yhat_beta.PS.AIC ,
    yhat_beta.SM.lasso ,
    yhat_beta.S.lasso ,
    yhat_beta.PS.lasso ,
    yhat_beta.lasso ,
    yhat_beta.enet ,
    yhat_beta.alasso ,
    yhat_beta.scad ,
    yhat_beta.ridge )
  #Calculate prediction errors of estimators
  PE_values <- c(FM=Prediction_Error(y_test_scale, yhat_beta.FM),
                 SM.BIC=Prediction_Error(y_test_scale, yhat_beta.SM.BIC),
                 S.BIC =Prediction_Error(y_test_scale, yhat_beta.S.BIC),
                 PS.BIC=Prediction_Error(y_test_scale, yhat_beta.PS.BIC),
                 SM.AIC=Prediction_Error(y_test_scale, yhat_beta.SM.AIC),
                 S.AIC =Prediction_Error(y_test_scale, yhat_beta.S.AIC),
                 PS.AIC=Prediction_Error(y_test_scale, yhat_beta.PS.AIC),
                 SM.lasso=Prediction_Error(y_test_scale, yhat_beta.SM.lasso),
                 S.lasso =Prediction_Error(y_test_scale, yhat_beta.S.lasso),
                 PS.lasso=Prediction_Error(y_test_scale, yhat_beta.PS.lasso),
                 LASSO = Prediction_Error( y_test_scale, yhat_beta.lasso),
                 ENET = Prediction_Error( y_test_scale, yhat_beta.enet),
                 ALASSO = Prediction_Error( y_test_scale, yhat_beta.alasso),
                 SCAD = Prediction_Error( y_test_scale , yhat_beta.scad ),
                 RIDGE = Prediction_Error( y_test_scale , yhat_beta.ridge)
  )
  PE_values_bootstrap[i,] <- PE_values
}
PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))
# Add row names
row_names <- c("Full␣Model","Submodel␣BIC","Shrinkage␣Estimator␣BIC",
"Positive␣Shrinkage␣Estimator␣BIC", "Submodel␣AIC","Shrinkage␣Estimator␣AIC",
"Positive␣Shrinkage␣Estimator␣AIC", "Submodel␣LASSO","Shrinkage␣Estimator␣LASSO",
"Positive␣Shrinkage␣Estimator␣LASSO", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
print_with_title <- function(df) {
  cat("Response␣Cryptocurrency:", Response_Coin, "\n\n")
  print(df) }
```

```
print_with_title(PE.result)

# Prediction model for Next one week
head(crypto_fridays_df)
crypto_fridays_nextweek_df <- crypto_fridays_df %>%
  select(
    Date,
    SP500_close_1WeekReturn,
    DowJones_close_1WeekReturn,
    NASDAQ_close_1WeekReturn,
    Russell2000_close_1WeekReturn,
    Nasdaq100_close_1WeekReturn,
    DJOilGas_close_1WeekReturn,
    DJGoldMining_close_1WeekReturn,
    XAU_USD_close_1WeekReturn,
    BitcoinClose_1WeekReturn,
    EthereumClose_1WeekReturn,
    EthereumClassicClose_1WeekReturn,
    TetherUSDtClose_1WeekReturn,
    XRPClose_1WeekReturn,
    CardanoClose_1WeekReturn,
    DogecoinClose_1WeekReturn,
    SP500_close_2WeekReturn,
    DowJones_close_2WeekReturn,
    NASDAQ_close_2WeekReturn,
    Russell2000_close_2WeekReturn,
    Nasdaq100_close_2WeekReturn,
    DJOilGas_close_2WeekReturn,
    DJGoldMining_close_2WeekReturn,
    XAU_USD_close_2WeekReturn,
    BitcoinClose_2WeekReturn,
    EthereumClose_2WeekReturn,
    EthereumClassicClose_2WeekReturn,
    TetherUSDtClose_2WeekReturn,
    XRPClose_2WeekReturn,
    CardanoClose_2WeekReturn,
    DogecoinClose_2WeekReturn
  )
head(crypto_fridays_nextweek_df)
```

```
# Assign each coin's return to Response_Coin, ONLY RUN ONE ROE for each coin model
Response_Coin = "BitcoinClose_1WeekReturn"
Response_Coin = "EthereumClose_1WeekReturn"
Response_Coin = "EthereumClassicClose_1WeekReturn"
Response_Coin = "TetherUSDtClose_1WeekReturn"
Response_Coin = "XRPClose_1WeekReturn"
Response_Coin = "CardanoClose_1WeekReturn"
Response_Coin = "DogecoinClose_1WeekReturn"

crypto_fridays_nextweek_model <- crypto_fridays_nextweek_df %>%
  arrange(Date) %>%
  mutate(y_Coin = lead(!!sym(Response_Coin)))
crypto_fridays_nextweek_model <- crypto_fridays_nextweek_model
[complete.cases(crypto_fridays_nextweek_model),]
y <- crypto_fridays_nextweek_model %>% dplyr :: select(y_Coin)%>%as.matrix()
X <- crypto_fridays_nextweek_model %>% dplyr :: select(-c(y_Coin,Date))%>%as.matrix()
X_intercept <-cbind(1,X)
raw_data <- data.frame(y,X)
p <- ncol(X)+1
head(raw_data)

# Fit Full model
full_names <- colnames(X)
xcount.FM <- c(full_names)
y_name <- colnames(y)
Formula_FM <- as.formula(paste(y_name, "␣~␣",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,raw_data)
summary(model_full)
vif(model_full)

#Submodel by BIC
step_model <- step(model_full, direction = "both",k = log(nrow(raw_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
full_names <- colnames(X)
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
Formula_SM.BIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM, collapse = "+")))
```

```
summary(step_model)
# If BIC choose no variable, use the below one
Formula_SM.BIC<- as.formula(paste(y_name, "␣~␣1"))

#Submodel by AIC
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.AIC,
collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y , alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
p2.lasso <- p-p1.lasso
xcount.SM.lasso <- c(sub_names.lasso)
Formula_SM.lasso <- as.formula(paste(y_name, "␣~␣",paste(xcount.SM.lasso,
collapse = "+")))
submodel_lasso <-lm(Formula_SM.lasso,raw_data)
summary(submodel_lasso)
# If LASSO choose no variable, use the below one
Formula_SM.lasso<- as.formula(paste(y_name, "␣~␣1"))

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)

# Set the seed for reproducibility
set.seed(1234)
```

```
for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(raw_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- raw_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(y_name) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-y_name) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(y_name) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-y_name) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
  df_train <- data.frame( y_train, X_train)
  #The full model fit based on train data
  fit_FM <-lm(Formula_FM,df_train)
  beta.FM <- fit_FM$coef
  beta.FM[is.na(beta.FM)] <- 0
  #The sub model fit based on train data
  fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
  beta.SM.BIC <- rep(0,p)
  beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
  beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
  fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
  beta.SM.AIC <- rep(0,p)
  beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
  beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
  fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
  beta.SM.lasso <- rep(0,p)
  beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
  lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE,
```

```
standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
```

```
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge
yhat_df <- data.frame(y_test_scale,yhat_beta.FM,yhat_beta.SM.BIC,
    yhat_beta.S.BIC ,
    yhat_beta.PS.BIC ,
    yhat_beta.SM.AIC ,
    yhat_beta.S.AIC ,
    yhat_beta.PS.AIC ,
    yhat_beta.SM.lasso ,
    yhat_beta.S.lasso ,
    yhat_beta.PS.lasso ,
    yhat_beta.lasso ,
    yhat_beta.enet ,
    yhat_beta.alasso ,
    yhat_beta.scad ,
    yhat_beta.ridge )
#Calculate prediction errors of estimators
PE_values <- c(FM=Prediction_Error(y_test_scale, yhat_beta.FM),
               SM.BIC=Prediction_Error(y_test_scale, yhat_beta.SM.BIC),
               S.BIC =Prediction_Error(y_test_scale, yhat_beta.S.BIC),
               PS.BIC=Prediction_Error(y_test_scale, yhat_beta.PS.BIC),
               SM.AIC=Prediction_Error(y_test_scale, yhat_beta.SM.AIC),
```

```
                S.AIC =Prediction_Error(y_test_scale, yhat_beta.S.AIC),
                PS.AIC=Prediction_Error(y_test_scale, yhat_beta.PS.AIC),
                SM.lasso=Prediction_Error(y_test_scale, yhat_beta.SM.lasso),
                S.lasso =Prediction_Error(y_test_scale, yhat_beta.S.lasso),
                PS.lasso=Prediction_Error(y_test_scale, yhat_beta.PS.lasso),
                LASSO = Prediction_Error( y_test_scale, yhat_beta.lasso),
                ENET = Prediction_Error( y_test_scale, yhat_beta.enet),
                ALASSO = Prediction_Error( y_test_scale, yhat_beta.alasso),
                SCAD = Prediction_Error( y_test_scale , yhat_beta.scad ),
                RIDGE = Prediction_Error( y_test_scale , yhat_beta.ridge)
  )
  PE_values_bootstrap[i,] <- PE_values
}
PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))

# Add row names
row_names <- c("Full␣Model","Submodel␣BIC","Shrinkage␣Estimator␣BIC",
"Positive␣Shrinkage␣Estimator␣BIC", "Submodel␣AIC","Shrinkage␣Estimator␣AIC",
"Positive␣Shrinkage␣Estimator␣AIC", "Submodel␣LASSO","Shrinkage␣Estimator␣LASSO",
"Positive␣Shrinkage␣Estimator␣LASSO", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
print_with_title <- function(df) {
  cat("Response␣Cryptocurrency:", Response_Coin, "\n\n")
  print(df) }
print_with_title(PE.result)

# Correlation Matrix on weekly data
crypto_weeklyReturn <- crypto_fridays_df %>%
  select(
    SP500_close_1WeekReturn,
    DowJones_close_1WeekReturn,
    NASDAQ_close_1WeekReturn,
    Russell2000_close_1WeekReturn,
    Nasdaq100_close_1WeekReturn,
    DJOilGas_close_1WeekReturn,
    DJGoldMining_close_1WeekReturn,
    XAU_USD_close_1WeekReturn,
    BitcoinClose_1WeekReturn,
```

```
    EthereumClose_1WeekReturn,
    EthereumClassicClose_1WeekReturn,
    TetherUSDtClose_1WeekReturn,
    XRPClose_1WeekReturn,
    CardanoClose_1WeekReturn,
    DogecoinClose_1WeekReturn,
    SP500_close_2WeekReturn,
    DowJones_close_2WeekReturn,
    NASDAQ_close_2WeekReturn,
    Russell2000_close_2WeekReturn,
    Nasdaq100_close_2WeekReturn,
    DJOilGas_close_2WeekReturn,
    DJGoldMining_close_2WeekReturn,
    XAU_USD_close_2WeekReturn,
    BitcoinClose_2WeekReturn,
    EthereumClose_2WeekReturn,
    EthereumClassicClose_2WeekReturn,
    TetherUSDtClose_2WeekReturn,
    XRPClose_2WeekReturn,
    CardanoClose_2WeekReturn,
    DogecoinClose_2WeekReturn,
    SP500_close_3WeekReturn,
    DowJones_close_3WeekReturn,
    NASDAQ_close_3WeekReturn,
    Russell2000_close_3WeekReturn,
    Nasdaq100_close_3WeekReturn,
    DJOilGas_close_3WeekReturn,
    DJGoldMining_close_3WeekReturn,
    XAU_USD_close_3WeekReturn,
    BitcoinClose_3WeekReturn,
    EthereumClose_3WeekReturn,
    EthereumClassicClose_3WeekReturn,
    TetherUSDtClose_3WeekReturn,
    XRPClose_3WeekReturn,
    CardanoClose_3WeekReturn,
    DogecoinClose_3WeekReturn
  )
head(crypto_weeklyReturn)

OneWeekReturn <- crypto_fridays_df %>%
```

```
  select(
    SP500_close_1WeekReturn,
    DowJones_close_1WeekReturn,
    NASDAQ_close_1WeekReturn,
    Russell2000_close_1WeekReturn,
    Nasdaq100_close_1WeekReturn,
    DJOilGas_close_1WeekReturn,
    DJGoldMining_close_1WeekReturn,
    XAU_USD_close_1WeekReturn,
    BitcoinClose_1WeekReturn,
    EthereumClose_1WeekReturn,
    EthereumClassicClose_1WeekReturn,
    TetherUSDtClose_1WeekReturn,
    XRPClose_1WeekReturn,
    CardanoClose_1WeekReturn,
    DogecoinClose_1WeekReturn
  )

TwoWeekReturn <- crypto_fridays_df %>%
  select(
    SP500_close_2WeekReturn,
    DowJones_close_2WeekReturn,
    NASDAQ_close_2WeekReturn,
    Russell2000_close_2WeekReturn,
    Nasdaq100_close_2WeekReturn,
    DJOilGas_close_2WeekReturn,
    DJGoldMining_close_2WeekReturn,
    XAU_USD_close_2WeekReturn,
    BitcoinClose_2WeekReturn,
    EthereumClose_2WeekReturn,
    EthereumClassicClose_2WeekReturn,
    TetherUSDtClose_2WeekReturn,
    XRPClose_2WeekReturn,
    CardanoClose_2WeekReturn,
    DogecoinClose_2WeekReturn
  )

# Set the graphical parameters to increase the plot size
par(oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0))
### Chatterjee Correlation Matrix for 1 weeks returns
```

```
correlation_df <- OneWeekReturn
#Compute the cross rank increment correlation coefficient xi
cor_xi <- xicor(correlation_df, ties = TRUE)
cor_xi<-cor_xi/cor_xi[1,1]
cor_xi
# Pearson Correlation Matrix
cor_xi <- cor(correlation_df)
rownames(cor_xi) <- gsub("_close_1WeekReturn|Close_1WeekReturn", "", rownames(cor_xi))
colnames(cor_xi) <- gsub("_close_1WeekReturn|Close_1WeekReturn", "", colnames(cor_xi))
corrplot(cor_xi,
        method = 'shade',
        addCoef.col = 'black',
        tl.col = "black",
        tl.cex = 0.8, # Adjust the text label size (larger size)
        number.cex = 0.6, # Adjust the correlation coefficients size
        mar = c(0, 0, 0, 0),
        addgrid.col = 'white')

### Chatterjee Correlation Matrix for 2 weeks returns
correlation_df <- TwoWeekReturn
#Compute the cross rank increment correlation coefficient xi
cor_xi <- xicor(correlation_df, ties = TRUE)
cor_xi<-cor_xi/cor_xi[1,1]
cor_xi
# Pearson Correlation Matrix
cor_xi <-cor(correlation_df)
rownames(cor_xi) <- gsub("_close_2WeekReturn|Close_2WeekReturn", "", rownames(cor_xi))
colnames(cor_xi) <- gsub("_close_2WeekReturn|Close_2WeekReturn", "", colnames(cor_xi))
corrplot(cor_xi,
        method = 'shade',
        addCoef.col = 'black',
        tl.col = "black",
        tl.cex = 0.8, # Adjust the text label size (larger size)
        number.cex = 0.6, # Adjust the correlation coefficients size
        mar = c(0, 0, 0, 0),
        addgrid.col = 'white')
```

D.2 Prediction Models with Principal Components

```
# Prediction model on Present Data
weeklyReturnPresent <- crypto_fridays_df %>%
  select(
    SP500_close_1WeekReturn,
    DowJones_close_1WeekReturn,
    NASDAQ_close_1WeekReturn,
    Russell2000_close_1WeekReturn,
    Nasdaq100_close_1WeekReturn,
    XAU_USD_close_1WeekReturn,
    DJOilGas_close_1WeekReturn,
    DJGoldMining_close_1WeekReturn,
    BitcoinClose_1WeekReturn,
    EthereumClose_1WeekReturn,
    EthereumClassicClose_1WeekReturn,
    TetherUSDtClose_1WeekReturn,
    XRPClose_1WeekReturn,
    CardanoClose_1WeekReturn,
    DogecoinClose_1WeekReturn
  )

# Select and perform PCA on stock indices and extract the first principal component
stock_indices <- weeklyReturnPresent %>%
  select(SP500_close_1WeekReturn, DowJones_close_1WeekReturn,
  NASDAQ_close_1WeekReturn, Russell2000_close_1WeekReturn, Nasdaq100_close_1WeekReturn)
stock_pca <- princomp(stock_indices, cor = FALSE)
stock_pc1 <- stock_pca$scores[, 1]
summary(stock_pca)

# Select and perform PCA on commodities and extract the first principal component
goldOil <- weeklyReturnPresent %>%
  select(XAU_USD_close_1WeekReturn, DJOilGas_close_1WeekReturn,
  DJGoldMining_close_1WeekReturn)
goldOil_pca <- princomp(goldOil, cor = FALSE)
goldOils_pc1 <- goldOil_pca$scores[, 1]
summary(goldOil_pca)

# Combine the first principal components with the cryptocurrency returns
cryptos <- weeklyReturnPresent %>%
  select(BitcoinClose_1WeekReturn, EthereumClose_1WeekReturn,
```

```
  EthereumClassicClose_1WeekReturn, TetherUSDtClose_1WeekReturn,
  XRPClose_1WeekReturn, CardanoClose_1WeekReturn, DogecoinClose_1WeekReturn)
weeklyReturnPresentPC <- data.frame(
  Stock_PC1 = stock_pc1,
  GoldOils_PC1 = goldOils_pc1,
  cryptos
)
head(weeklyReturnPresentPC)

# List of response variables
cryptocurrencies <- c("BitcoinClose_1WeekReturn", "EthereumClose_1WeekReturn",
"EthereumClassicClose_1WeekReturn", "TetherUSDtClose_1WeekReturn",
"XRPClose_1WeekReturn", "CardanoClose_1WeekReturn", "DogecoinClose_1WeekReturn")

# Loop through each cryptocurrency and perform linear regression
models <- list()
for (crypto in cryptocurrencies) {
  # Create the formula for the linear regression
  formula <- as.formula(paste(crypto, "~␣."))
  # Perform the linear regression
  model <- lm(formula, data = weeklyReturnPresentPC)
  # Store the model in the list
  models[[crypto]] <- model
  # Print summaries of the models
  cat("\nLinear␣regression␣model␣for", crypto, ":\n")
  print(summary(model))
  print(vif(model))
}

# Bootstrap samples and Prediction error
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}
# Number of bootstrap samples
B <- 250
set.seed(1234)
# Store prediction error results
PE_results <- list()

for (crypto in cryptocurrencies) {
  cat("\nProcessing:", crypto, "\n")
  PE_values_bootstrap <- numeric(B)
```

```
  for (i in 1:B) {
    # Bootstrap sample
    bootstrap_indices <- sample(nrow(weeklyReturnPresentPC), replace = TRUE)
    bootstrap_sample <- weeklyReturnPresentPC[bootstrap_indices, ]
    # Split into train/test
    set.seed(i)
    split <- sample.split(bootstrap_sample[[crypto]], SplitRatio = 0.75)
    train_set <- subset(bootstrap_sample, split == TRUE)
    test_set <- subset(bootstrap_sample, split == FALSE)
    # Fit model on train set
    formula <- as.formula(paste(crypto, "~ ."))
    model <- lm(formula, data = train_set)
    # Predict on test set
    y_test <- test_set[[crypto]]
    yhat <- predict(model, newdata = test_set)
    # Compute prediction error
    PE_values_bootstrap[i] <- Prediction_Error(y_test, yhat)
  }
  # Store average and std dev of prediction error
  PE_avg <- mean(PE_values_bootstrap)
  PE_std <- sd(PE_values_bootstrap)
  print(PE_avg)
  print(PE_std)
  PE_results[[crypto]] <- data.frame(PE_avg = PE_avg, PE_std = PE_std)
}
# Combine results into a single data frame
PE_summary <- do.call(rbind, PE_results)
rownames(PE_summary) <- cryptocurrencies
print(PE_summary)

# Prediction model on Next week
# List of columns to exclude
columns_to_exclude <- c("Weekday", "SP500_close", "DowJones_close",
"NASDAQ_close", "Russell2000_close", "XAU_USD_close", "Nasdaq100_close",
"DJOilGas_close", "DJGoldMining_close", "BitcoinClose", "EthereumClose",
"EthereumClassicClose", "TetherUSDtClose", "XRPClose", "CardanoClose", "DogecoinClose")
weeklyReturnNextWeek <- crypto_fridays_df %>%
  select(-all_of(columns_to_exclude))
head(weeklyReturnNextWeek)
# Order the dataframe by date
```

```
weeklyReturnNextWeek <- weeklyReturnNextWeek %>%
  arrange(Date)

# Perform PCA on stock indices for 1-week, 2-weeks
stock_indices_1week <- weeklyReturnNextWeek %>%
  select(SP500_close_1WeekReturn, DowJones_close_1WeekReturn, NASDAQ_close_1WeekReturn,
  Russell2000_close_1WeekReturn, Nasdaq100_close_1WeekReturn)

stock_indices_2week <- weeklyReturnNextWeek %>%
  select(SP500_close_2WeekReturn, DowJones_close_2WeekReturn, NASDAQ_close_2WeekReturn,
  Russell2000_close_2WeekReturn, Nasdaq100_close_2WeekReturn)

stock_pca_1week <- princomp(stock_indices_1week, cor = FALSE)
stock_pc1_1week <- stock_pca_1week$scores[, 1]
summary(stock_pca_1week)

stock_pca_2week <- princomp(stock_indices_2week, cor = FALSE)
stock_pc1_2week <- stock_pca_2week$scores[, 1]
summary(stock_pca_2week)

# Perform PCA on GoldOil for 1-week, 2-weeks
GoldOil_1week <- weeklyReturnNextWeek %>%
  select(XAU_USD_close_1WeekReturn, DJOilGas_close_1WeekReturn,
  DJGoldMining_close_1WeekReturn)

GoldOil_2week <- weeklyReturnNextWeek %>%
  select(XAU_USD_close_2WeekReturn, DJOilGas_close_2WeekReturn,
  DJGoldMining_close_2WeekReturn)

GoldOil_pca_1week <- princomp(GoldOil_1week, cor = FALSE)
GoldOil_pc1_1week <- GoldOil_pca_1week$scores[, 1]
summary(GoldOil_pca_1week)

GoldOil_pca_2week <- princomp(GoldOil_2week, cor = FALSE)
GoldOil_pc1_2week <- GoldOil_pca_2week$scores[, 1]
summary(GoldOil_pca_2week)

cryptos <- weeklyReturnNextWeek %>%
  select(BitcoinClose_1WeekReturn, EthereumClose_1WeekReturn,
  EthereumClassicClose_1WeekReturn, TetherUSDtClose_1WeekReturn,
```

```
  XRPClose_1WeekReturn, CardanoClose_1WeekReturn, DogecoinClose_1WeekReturn,
  BitcoinClose_2WeekReturn, EthereumClose_2WeekReturn,
  EthereumClassicClose_2WeekReturn, TetherUSDtClose_2WeekReturn,
  XRPClose_2WeekReturn, CardanoClose_2WeekReturn, DogecoinClose_2WeekReturn)

# Add the first principal components to the dataframe
weeklyReturnNextWeekPC <- data.frame(
  Stock_PC1_1Week = stock_pc1_1week,
  GoldOil_PC1_1Week = GoldOil_pc1_1week,
  Stock_PC1_2Week = stock_pc1_2week,
  GoldOil_PC1_2Week = GoldOil_pc1_2week,
  cryptos
)
head(weeklyReturnNextWeekPC)

# List of response variables
cryptocurrencies <- c("BitcoinClose_1WeekReturn", "EthereumClose_1WeekReturn",
"EthereumClassicClose_1WeekReturn", "TetherUSDtClose_1WeekReturn",
"XRPClose_1WeekReturn", "CardanoClose_1WeekReturn", "DogecoinClose_1WeekReturn")

# Create a copy of the dataframe to include lead response variables
lead_df <- weeklyReturnNextWeekPC
# Create lead response variables for each cryptocurrency
for (crypto in cryptocurrencies) {
  lead_df[[paste0(crypto, "_nextWeek")]] <- dplyr::lead(lead_df[[crypto]], 1)
}

# Drop the last row with NA in lead response variables
lead_df <- lead_df[!is.na(lead_df[[paste0(cryptocurrencies[1], "_nextWeek")]]), ]
head(lead_df)

# Define the predictors
predictors <- c("Stock_PC1_1Week", "GoldOil_PC1_1Week",
                "Stock_PC1_2Week", "GoldOil_PC1_2Week",
                "BitcoinClose_1WeekReturn", "EthereumClose_1WeekReturn",
                "EthereumClassicClose_1WeekReturn", "TetherUSDtClose_1WeekReturn",
                "XRPClose_1WeekReturn", "CardanoClose_1WeekReturn",
                "DogecoinClose_1WeekReturn", "BitcoinClose_2WeekReturn",
                "EthereumClose_2WeekReturn", "EthereumClassicClose_2WeekReturn",
                "TetherUSDtClose_2WeekReturn", "XRPClose_2WeekReturn",
```

```
              "CardanoClose_2WeekReturn", "DogecoinClose_2WeekReturn")

# Loop through each cryptocurrency and perform linear regression
models <- list()
for (crypto in cryptocurrencies) {
  # Create the formula for the linear regression with lead response variable
  formula <- as.formula(paste(paste0(crypto, "_nextWeek"), "~",
  paste(predictors, collapse = "␣+␣")))
  # Perform the linear regression
  model <- lm(formula, data = lead_df)
  # Store the model in the list
  models[[crypto]] <- model
  # Print summary of the model
  cat("\nLinear␣regression␣model␣for", paste0(crypto, "_nextWeek"), ":\n")
  print(summary(models[[crypto]]))
  print(vif(models[[crypto]]))
}

# Bootstrap samples and Prediction error
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}
# Number of bootstrap samples
B <- 250
set.seed(1234)
# Store prediction error results
PE_results <- list()
for (crypto in cryptocurrencies) {
  cat("\nProcessing:", crypto, "\n")
  response <- paste0(crypto, "_nextWeek")
  PE_values_bootstrap <- numeric(B)
  for (i in 1:B) {
    # Bootstrap sample
    bootstrap_indices <- sample(nrow(lead_df), replace = TRUE)
    bootstrap_sample <- lead_df[bootstrap_indices, ]
    # Split into train/test
    set.seed(i)
    split <- sample.split(bootstrap_sample[[response]], SplitRatio = 0.75)
    train_set <- subset(bootstrap_sample, split == TRUE)
    test_set <- subset(bootstrap_sample, split == FALSE)
    # Fit model on train set
    formula <- as.formula(paste(paste0(crypto, "_nextWeek"), "~",
```

```
    paste(predictors, collapse = " + ")))
    model <- lm(formula, data = train_set)
    # Predict on test set
    y_test <- test_set[[response]]
    yhat <- predict(model, newdata = test_set)
    # Compute prediction error
    PE_values_bootstrap[i] <- Prediction_Error(y_test, yhat)
  }
  # Store average and std dev of prediction error
  PE_avg <- mean(PE_values_bootstrap)
  PE_std <- sd(PE_values_bootstrap)
  print(PE_avg)
  print(PE_std)
  PE_results[[response]] <- data.frame(PE_avg = PE_avg, PE_std = PE_std)
}
# Combine results into a single data frame
PE_summary <- do.call(rbind, PE_results)
rownames(PE_summary) <- cryptocurrencies
print(PE_summary)
```

APPENDIX E

SOURCE CODE OF CHAPTER SIX

```
# Correlation Matrix
cor(cryptocurrency_df$BitcoinClose,cryptocurrency_df$EthereumClose)
cor(cryptocurrency_df$EthereumClose,cryptocurrency_df$EthereumClassicClose)
cor(cryptocurrency_df$BitcoinClose,cryptocurrency_df$EthereumClassicClose)
crypto_log_df <- data.frame(
  Date = cryptocurrency_df$Date,
  LogBitcoin = log(cryptocurrency_df$BitcoinClose) ,
  LogEthereum = log(cryptocurrency_df$EthereumClose) ,
  LogEthereumCls = log(cryptocurrency_df$EthereumClassicClose)
)
head(crypto_log_df)
cor(crypto_log_df$LogEthereum,crypto_log_df$LogEthereumCls)
cor(crypto_log_df$LogBitcoin,crypto_log_df$LogEthereumCls)
cor(crypto_log_df$LogBitcoin,crypto_log_df$LogEthereum)

# Plot the data
ggplot(data = crypto_log_df, aes(x = Date)) +
  geom_line(aes(y = LogBitcoin, color = "Bitcoin", linetype = "Bitcoin")) +
  geom_line(aes(y = LogEthereum, color = "Ethereum", linetype = "Ethereum")) +
  labs( #title = "Logarithmic Prices of Bitcoin and Ethereum Over Time",
    x = "Date",
    y = "Logarithmic␣Price",
    color = "Cryptocurrency",
    linetype = "Cryptocurrency") +
  theme_minimal() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"), # Add y-axis line
        legend.position = "top")

ggplot(data = crypto_log_df, aes(x = Date)) +
  geom_line(aes(y = LogEthereum, color = "Ethereum",linetype = "Ethereum")) +
  geom_line(aes(y = LogEthereumCls, color = "EthereumClassic",
  linetype = "EthereumClassic")) +
  labs(#title = "Logarithmic Prices of Ethereum and Ethereum Classic Over Time",
    x = "Date",
    y = "Logarithmic␣Price",
    color = "Cryptocurrency",
```

```
    linetype = "Cryptocurrency") +
  theme_minimal() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"), # Add y-axis line
        legend.position = "top")

ggplot(data = crypto_log_df, aes(x = Date)) +
  geom_line(aes(y = LogBitcoin, color = "Bitcoin",linetype = "Bitcoin")) +
  geom_line(aes(y = LogEthereumCls, color = "EthereumClassic",
  linetype = "EthereumClassic")) +
  labs(#title = "Logarithmic Prices of Bitcoin and Ethereum Classic Over Time",
    x = "Date",
    y = "Logarithmic␣Price",
    color = "Cryptocurrency",
    linetype = "Cryptocurrency") +
  theme_minimal() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"), # Add y-axis line
        legend.position = "top")

# Cointegration test
library(urca)
library(vars)
library(tseries)
library(fUnitRoots)
# Least-squares estimation: Bitcoin and Ethereum
# Fit the linear model and Extract residuals
cointegration_reg <- lm(LogEthereum ~ LogBitcoin, data = crypto_log_df)
summary(cointegration_reg)
residuals <- residuals(cointegration_reg)

# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals␣Over␣Time", x = "Date", y = "Residuals") +
  theme_minimal()
```

```
# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The number of lags selected by AIC is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
adfTest(residuals,lags=7,type=c("nc"))
adfTest(residuals,lags=1,type=c("nc"))
coint_test <- ur.df(residuals, type = "none", lags = 7)
summary(coint_test)

# Fit the linear model and Extract residuals
cointegration_reg <- lm(LogBitcoin ~ LogEthereum, data = crypto_log_df)
summary(cointegration_reg)
residuals <- residuals(cointegration_reg)

# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

## Least-squares estimation: Ethereum and Ethereum Classic
# Fit the linear model and Extract residuals
cointegration_reg <- lm(LogEthereum ~ LogEthereumCls, data = crypto_log_df)
summary(cointegration_reg)
```

```
residuals <- residuals(cointegration_reg)

# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The number of lags selected by AIC is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
adfTest(residuals,lags=1,type=c("nc"))

## Least-squares estimation: Bitcoin and Ethereum Classic
# Fit the linear model and Extract residuals
cointegration_reg <- lm(LogBitcoin ~ LogEthereumCls, data = crypto_log_df)
summary(cointegration_reg)
residuals <- residuals(cointegration_reg)

# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)
```

```
# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The␣number␣of␣lags␣selected␣by␣AIC␣is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
adfTest(residuals,lags=1,type=c("nc"))

## Least-squares estimation: Bitcoin and Ethereum, Ethereum Classic
# Fit the linear model and Extract residuals
cointegration_reg <- lm(LogBitcoin ~ LogEthereum + LogEthereumCls,
data = crypto_log_df)
summary(cointegration_reg)
residuals <- residuals(cointegration_reg)

# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals␣Over␣Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF␣of␣Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The␣number␣of␣lags␣selected␣by␣AIC␣is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
adfTest(residuals,lags=1,type=c("nc"))

####### Trading strategy #######
# Fit the linear model and get spread series
cointegration_reg <- lm(LogEthereum ~ LogBitcoin, data = crypto_log_df)
summary(cointegration_reg)
```

```
crypto_log_df$Wt <- crypto_log_df$LogEthereum - 1.289980*crypto_log_df$LogBitcoin
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
#delta <-0.4
#delta <-0.4/2
delta <-0.4*(3/4)

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Bitcoin_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)
```

```
# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
crypto_log_df$Wt <- crypto_log_df$LogEthereum - 1.289980*crypto_log_df$LogBitcoin

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long LogEthereum, Short LogBitcoin
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
```

```
view(crypto_log_df)
crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogBitcoin, LogEthereum, Wt,position,return)

# Calculate cumulative returns
crypto_log_df$cumulative_return <- cumsum(crypto_log_df$return)

# Plot the cumulative returns
ggplot(data = crypto_log_df, aes(x = Date, y = cumulative_return)) +
  geom_line() +
  labs(title = "Pair␣Trading␣Strategy␣Cumulative␣Returns", x = "Date",
  y = "Cumulative␣Return") +
  theme_minimal()
Total_logReturn = sum(crypto_log_df$return)
Total_logReturn

trade_df <-crypto_log_df[crypto_log_df$position != 0, ]
# Select required columns and add Bitcoin and Ethereum columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin, LogEthereum, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    Ethereum = exp(LogEthereum)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Ethereum[i-1])*
    (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])) -
      (1.289980 *1000/trade_df$Bitcoin[i-1])*(trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])
  }
}
trade_df

### Trading strategy 2: reverse the variables ###
# Fit the linear model and get spread series
cointegration_reg <- lm(LogBitcoin ~ LogEthereum, data = crypto_log_df)
```

```
summary(cointegration_reg)

residuals <- residuals(cointegration_reg)
# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The number of lags selected by AIC is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
adfTest(residuals,lags=1,type=c("nc"))

crypto_log_df$Wt <- crypto_log_df$LogBitcoin - 0.682213*crypto_log_df$LogEthereum
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
#delta <-0.3
#delta <-0.3/2
delta <-0.3*(3/4)

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread Over Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
```

```
  select(Date, LogBitcoin)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Bitcoin_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
crypto_log_df$Wt <- crypto_log_df$LogBitcoin - 0.682213*crypto_log_df$LogEthereum

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA
# Loop through the data to identify entry and exit points
```

```
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogBitcoin, LogEthereum, Wt,position,return)

# Calculate cumulative returns
crypto_log_df$cumulative_return <- cumsum(crypto_log_df$return)
# Plot the cumulative returns
ggplot(data = crypto_log_df, aes(x = Date, y = cumulative_return)) +
  geom_line() +
  labs(title = "Pair␣Trading␣Strategy␣Cumulative␣Returns", x = "Date",
  y = "Cumulative␣Return") +
  theme_minimal()

Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns and add Bitcoin and Ethereum columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin, LogEthereum, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    Ethereum = exp(LogEthereum)
  )
```

```
# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Bitcoin[i-1])*
    (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])) -
      (0.682213 *1000/trade_df$Ethereum[i-1])
      *(trade_df$Ethereum[i]-trade_df$Ethereum[i-1])
  }
}
trade_df

### Trading strategy 3: train and test ###
head(crypto_log_df)
# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Bitcoin_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
```

```
crypto_log_df <- crypto_log_df %>%
  arrange(Date)

# Extract year and count rows by year
count_by_year <- crypto_log_df %>%
  mutate(
    year = format(Date, "%Y"),
    month = format(Date, "%m")
  ) %>%
  group_by(year) %>%
  summarize(count = n(), .groups = 'drop')
print(count_by_year)

#Filter the dataframe for the training set
train_set <- crypto_log_df %>%
  filter(Date >= as.Date("2018-01-01") & Date <= as.Date("2020-12-31"))

count_by_year <- train_set %>%
  mutate(year = format(Date, "%Y") ) %>%
  group_by(year) %>%
  summarize(count = n(), .groups = 'drop')
print(count_by_year)

# Fit the linear model and get spread series
cointegration_reg <- lm(LogBitcoin ~ LogEthereum, data = train_set)
summary(cointegration_reg)

residuals <- residuals(cointegration_reg)
# Plot ACF of residuals
acf(residuals, main = "ACF␣of␣Residuals")

# Perform unit root test on the residuals
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)

# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The␣number␣of␣lags␣selected␣by␣AIC␣is:", selected_lags, "\n")
m1=ar(diff(residuals),method='mle')
m1$order
```

```
adfTest(residuals,lags=1,type=c("nc"))
Trade_beta<-coef(cointegration_reg)["LogEthereum"]
train_set$Wt <- train_set$LogBitcoin - Trade_beta*train_set$LogEthereum
mean_wt <- mean(train_set$Wt)
sd_wt <- sd(train_set$Wt)
#delta <-0.31
delta <-0.31

# Create the plot and add horizontal lines
ggplot(data = train_set, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()
crypto_log_df$Wt <- crypto_log_df$LogBitcoin - Trade_beta*crypto_log_df$LogEthereum

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position
    crypto_log_df$position[i] <- 1
```

```
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns and add Bitcoin and Ethereum columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin, LogEthereum, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    Ethereum = exp(LogEthereum)
  )

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Bitcoin[i-1])*
    (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])) -
      (Trade_beta *1000/trade_df$Ethereum[i-1])*
      (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])
  }
}
trade_df

### Pair trading analysis on Ethereum and Ethereum Classic ###
crypto_log_df <- data.frame(
  Date = cryptocurrency_df$Date,
  LogBitcoin = log(cryptocurrency_df$BitcoinClose) ,
  LogEthereum = log(cryptocurrency_df$EthereumClose) ,
  LogEthereumCls = log(cryptocurrency_df$EthereumClassicClose)
```

```
)
head(crypto_log_df)
cor(crypto_log_df$LogEthereum,crypto_log_df$LogEthereumCls)

# Plot the data
ggplot(data = crypto_log_df, aes(x = Date)) +
  geom_line(aes(y = LogEthereum, color = "LogEthereum")) +
  geom_line(aes(y = LogEthereumCls, color = "LogEthereumClassic")) +
  labs(title = "Logarithmic Prices of Ethereum and Ethereum Classic Over Time",
       x = "Date",
       y = "Log Price",
       color = "Cryptocurrency") +
  theme_minimal()

### Trading strategy ####
# Fit the linear model and get spread series
cointegration_reg <- lm(LogEthereum ~ LogEthereumCls, data = crypto_log_df)
summary(cointegration_reg)
# Test cointegration
residuals <- residuals(cointegration_reg)
# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
library(urca)
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)
# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The number of lags selected by AIC is:", selected_lags, "\n")

crypto_log_df$Wt <- crypto_log_df$LogEthereum - 1.27403*crypto_log_df$LogEthereumCls
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
```

```
delta <-0.55

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Ethereumcl_New <- read.csv("ethereumclassic_new.csv", header = TRUE)
Ethereumcl_New$Date <- as.Date(Ethereumcl_New$..Date, format = "%m/%d/%Y")
Ethereumcl_New$LogEthereumCls <- log(Ethereumcl_New$Close)
Ethereumcl_New_filtered <- Ethereumcl_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereumCls)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Ethereumcl_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
crypto_log_df$Wt <- crypto_log_df$LogEthereum - 1.27403*crypto_log_df$LogEthereumCls
```

```
# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long Ethereum, Short Ethereum Classic
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogEthereum, LogEthereumCls, Wt,position,return)
Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
```

```
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns
trade_df <- trade_df %>%
  select(Date, LogEthereum,LogEthereumCls, position) %>%
  mutate(
    Ethereum = exp(LogEthereum),
    EthereumCls = exp(LogEthereumCls)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Ethereum[i-1])*
    (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])) -
      (1.27403 *1000/trade_df$EthereumCls[i-1])*
      (trade_df$EthereumCls[i]-trade_df$EthereumCls[i-1])
  }
}
trade_df

# Trading strategy 2: reverse the variables
# Fit the linear model and get spread series
cointegration_reg <- lm(LogEthereumCls ~ LogEthereum, data = crypto_log_df)
summary(cointegration_reg)
#Test cointegration
residuals <- residuals(cointegration_reg)
# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals␣Over␣Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF␣of␣Residuals")

# Perform unit root test on the residuals
library(urca)
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
```

```
summary(coint_test)
# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The␣number␣of␣lags␣selected␣by␣AIC␣is:", selected_lags, "\n")
crypto_log_df$Wt <- crypto_log_df$LogEthereumCls - 0.609331*crypto_log_df$LogEthereum
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
delta <-0.4

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess EthereumCls data
Ethereumcl_New <- read.csv("ethereumclassic_new.csv", header = TRUE)
Ethereumcl_New$Date <- as.Date(Ethereumcl_New$..Date, format = "%m/%d/%Y")
Ethereumcl_New$LogEthereumCls <- log(Ethereumcl_New$Close)
Ethereumcl_New_filtered <- Ethereumcl_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereumCls)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Ethereumcl_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
```

```
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
crypto_log_df$Wt <- crypto_log_df$LogEthereumCls - 0.609331*crypto_log_df$LogEthereum

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

#trading strategy is as follows:
# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position
    crypto_log_df$position[i] <- 1
```

```
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}

crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogEthereum, LogEthereumCls, Wt,position,return)

Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns
trade_df <- trade_df %>%
  select(Date, LogEthereum,LogEthereumCls, position) %>%
  mutate(
    Ethereum = exp(LogEthereum),
    EthereumCls = exp(LogEthereumCls)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$EthereumCls[i-1])*
    (trade_df$EthereumCls[i]-trade_df$EthereumCls[i-1])) -
      (0.609331*1000/trade_df$Ethereum[i-1])*
      (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])
  }
}
trade_df
```

```
#### Pair trading analysis on Bitcoin and Ethereum Classic ###
crypto_log_df <- data.frame(
  Date = cryptocurrency_df$Date,
  LogBitcoin = log(cryptocurrency_df$BitcoinClose) ,
  LogEthereumCls = log(cryptocurrency_df$EthereumClassicClose)
)
head(crypto_log_df)
cor(crypto_log_df$LogBitcoin,crypto_log_df$LogEthereumCls)

# Plot the data
ggplot(data = crypto_log_df, aes(x = Date)) +
  geom_line(aes(y = LogBitcoin, color = "LogBitcoin")) +
  geom_line(aes(y = LogEthereumCls, color = "LogEthereumClassic")) +
  labs(title = "Logarithmic␣Prices␣of␣Bitcoin␣and␣Ethereum␣Classic␣Over␣Time",
       x = "Date",
       y = "Log␣Price",
       color = "Cryptocurrency") +
  theme_minimal()

# Trading strategy
# Fit the linear model and get spread series
cointegration_reg <- lm(LogBitcoin ~ LogEthereumCls, data = crypto_log_df)
summary(cointegration_reg)

# Test cointegration
residuals <- residuals(cointegration_reg)
# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals␣Over␣Time", x = "Date", y = "Residuals") +
  theme_minimal()

# Plot ACF of residuals
acf(residuals, main = "ACF␣of␣Residuals")

# Perform unit root test on the residuals
library(urca)
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)
```

```
# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The␣number␣of␣lags␣selected␣by␣AIC␣is:", selected_lags, "\n")
crypto_log_df$Wt <- crypto_log_df$LogBitcoin - 0.78453*crypto_log_df$LogEthereumCls
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
delta <-0.56

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)

# Read and preprocess Ethereum classic data
Ethereumcl_New <- read.csv("ethereumclassic_new.csv", header = TRUE)
Ethereumcl_New$Date <- as.Date(Ethereumcl_New$..Date, format = "%m/%d/%Y")
Ethereumcl_New$LogEthereumCls <- log(Ethereumcl_New$Close)
Ethereumcl_New_filtered <- Ethereumcl_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereumCls)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Ethereumcl_New_filtered %>%
  full_join(Bitcoin_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
```

```
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
crypto_log_df$Wt <- crypto_log_df$LogBitcoin - 0.78453*crypto_log_df$LogEthereumCls
# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

#trading strategy is as follows:
# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long Bitcoin, Short Ethereum Classic
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
```

```
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin,LogEthereumCls, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    EthereumCls = exp(LogEthereumCls)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Bitcoin[i-1])*
    (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])) -
      (0.78453 *1000/trade_df$EthereumCls[i-1])*
      (trade_df$EthereumCls[i]-trade_df$EthereumCls[i-1])
  }
}
trade_df
# Reverse the variables
crypto_log_df <- data.frame(
  Date = cryptocurrency_df$Date,
  LogBitcoin = log(cryptocurrency_df$BitcoinClose) ,
  #LogEthereum = log(cryptocurrency_df$EthereumClose) ,
  LogEthereumCls = log(cryptocurrency_df$EthereumClassicClose)
```

```
)
head(crypto_log_df)

# Trading strategy
# Fit the linear model and get spread series
cointegration_reg <- lm(LogEthereumCls ~ LogBitcoin, data = crypto_log_df)
summary(cointegration_reg)
## Test cointegration
residuals <- residuals(cointegration_reg)
# Plot residuals over time
ggplot(data = crypto_log_df, aes(x = Date, y = residuals)) +
  geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residuals") +
  theme_minimal()
# Plot ACF of residuals
acf(residuals, main = "ACF of Residuals")

# Perform unit root test on the residuals
library(urca)
coint_test <- ur.df(residuals, type = "none", selectlags = "AIC")
summary(coint_test)
# Extract the selected number of lags
selected_lags <- coint_test@lags
cat("The number of lags selected by AIC is:", selected_lags, "\n")
crypto_log_df$Wt <- crypto_log_df$LogEthereumCls - 0.70949*crypto_log_df$LogBitcoin
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
delta <-0.54

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread Over Time", x = "Date", y = "Wt") +
  theme_minimal()

# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
```

```
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)
# Read and preprocess Ethereum classic data
Ethereumcl_New <- read.csv("ethereumclassic_new.csv", header = TRUE)
Ethereumcl_New$Date <- as.Date(Ethereumcl_New$..Date, format = "%m/%d/%Y")
Ethereumcl_New$LogEthereumCls <- log(Ethereumcl_New$Close)
Ethereumcl_New_filtered <- Ethereumcl_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereumCls)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Ethereumcl_New_filtered %>%
  full_join(Bitcoin_New_filtered, by = "Date")
# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)
# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)

crypto_log_df$Wt <- crypto_log_df$LogEthereumCls - 0.70949*crypto_log_df$LogBitcoin
#view(crypto_log_df)

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dotted") +
```

```
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs( x = "Date", y = "Spread␣(Wt)") +
  theme_minimal()

#trading strategy is as follows:
# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0
# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA
# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long Bitcoin, Short Ethereum Classic
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin,LogEthereumCls, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    EthereumCls = exp(LogEthereumCls)
```

```
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$EthereumCls[i-1])*
    (trade_df$EthereumCls[i]-trade_df$EthereumCls[i-1])) -
      (0.70949 *1000/trade_df$Bitcoin[i-1])*
      (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])
  }
}
trade_df
## Orthogonal regression ###
#(1) LogEthereum ~ LogBitcoin
# Fit the ordinary least squares (OLS) regression
OLS_reg <- lm(LogEthereum ~ LogBitcoin, data = crypto_log_df)
summary(OLS_reg)
OLS_reg

library(pracma)
# Orthogonal Distance Regression (ODR, total least squares)
odr <- odregress(crypto_log_df$LogBitcoin, crypto_log_df$LogEthereum)
odr$coeff
odr$ssq

# 1.R-square based on the ssq: sum of squares of orthogonal distances
r_squared_odreg <- function(object, y) {
  ss_tot <- sum((y - mean(y))^2)
  1 - object$ssq/ss_tot
}
r_squared_odreg(odr, crypto_log_df$LogEthereum)

# 2.R-square based on the vertical line
r_squared_odreg2 <- function(object, y) {
  ss_tot <- sum((y - mean(y))^2)
  ss_res <-sum((y - object$fitted)^2)
  1 - ss_res/ss_tot
}
r_squared_odreg2(odr, crypto_log_df$LogEthereum)
```

```
# Visual Comparison
# Plot data points
plot(crypto_log_df$LogBitcoin, crypto_log_df$LogEthereum,
     #main = "LogEthereum vs LogBitcoin with OLS and ODR Lines",
     xlab = "LogBitcoin", ylab = "LogEthereum", pch = 19, col = "black",cex=0.4)

# Add OLS regression line
abline(OLS_reg, col = "red", lwd = 2)

# Add ODR regression line
odr_slope <- odr$coeff[1]
odr_intercept <- odr$coeff[2]
abline(a = odr_intercept, b = odr_slope, col = "blue", lwd = 2, lty = 2)
legend("topleft", legend = c("OLS␣line", "ODR␣line"), col = c("red", "blue"),
lty = c(1, 2), lwd = 2)

#(2) LogBitcoin ~ LogEthereum
# Orthogonal Distance Regression
odr2 <- odregress(crypto_log_df$LogEthereum, crypto_log_df$LogBitcoin )
odr2$coeff

### Pair Trading Strategy ###
# ODR model: LogEthereum = -7.024313 + 1.403245 LogBitcoin.
# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)
```

```
# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Bitcoin_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
head(crypto_log_df)
crypto_log_df$Wt <- crypto_log_df$LogEthereum - 1.403245*crypto_log_df$LogBitcoin
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
delta <- 0.46

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

#trading strategy is as follows:
# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0
# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long LogEthereum, Short LogBitcoin
    crypto_log_df$position[i] <- 1
```

```
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}

crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogBitcoin, LogEthereum, Wt,position,return)

# Calculate cumulative returns
crypto_log_df$cumulative_return <- cumsum(crypto_log_df$return)
Total_logReturn = sum(crypto_log_df$return)
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]
# Select required columns and add Bitcoin and Ethereum columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin, LogEthereum, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    Ethereum = exp(LogEthereum)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Ethereum[i-1])*
    (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])) -
      (1.403245 *1000/trade_df$Bitcoin[i-1])*
      (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])
  }
}
```

```
trade_df

### Pair Trading Strategy - Reversed order###
# ODR model: LogBitcoin = 5.0057639+ 0.7126339 LogEthereum.
# Include out-of-sample period data: 2024-6-18 to 2025-2-27
# Read and preprocess Bitcoin data
Bitcoin_New <- read.csv("bitcoin_new.csv", header = TRUE)
Bitcoin_New$Date <- as.Date(Bitcoin_New$..Date, format = "%m/%d/%Y")
Bitcoin_New$LogBitcoin <- log(Bitcoin_New$Close)
Bitcoin_New_filtered <- Bitcoin_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogBitcoin)

# Read and preprocess Ethereum data
Ethereum_New <- read.csv("ethereum_new.csv", header = TRUE)
Ethereum_New$Date <- as.Date(Ethereum_New$..Date, format = "%m/%d/%Y")
Ethereum_New$LogEthereum <- log(Ethereum_New$Close)
Ethereum_New_filtered <- Ethereum_New %>%
  filter(Date >= as.Date("2024-06-18") & Date <= as.Date("2025-02-27")) %>%
  select(Date, LogEthereum)

# Merge the filtered Bitcoin and Ethereum data
merged_new_data <- Bitcoin_New_filtered %>%
  full_join(Ethereum_New_filtered, by = "Date")

# Bind the new data to the existing crypto_log_df
crypto_log_df <- crypto_log_df %>%
  bind_rows(merged_new_data)

# Arrange the dataframe by Date to maintain chronological order
crypto_log_df <- crypto_log_df %>%
  arrange(Date)
head(crypto_log_df)
crypto_log_df$Wt <- crypto_log_df$LogBitcoin - 0.7126339*crypto_log_df$LogEthereum
mean_wt <- mean(crypto_log_df$Wt)
sd_wt <- sd(crypto_log_df$Wt)
delta <- 0.33

# Create the plot and add horizontal lines
ggplot(data = crypto_log_df, aes(x = Date, y = Wt)) +
```

```
  geom_line() +
  geom_hline(yintercept = mean_wt, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = mean_wt + delta, color = "red", linetype = "dashed") +
  geom_hline(yintercept = mean_wt - delta, color = "red", linetype = "dashed") +
  labs(title = "Spread␣Over␣Time", x = "Date", y = "Wt") +
  theme_minimal()

#trading strategy is as follows:
# Initialize columns for positions and returns
crypto_log_df$position <- 0
crypto_log_df$return <- 0

# Initialize a flag to track if we are in a position
in_position <- FALSE
entry_index <- NA

# Loop through the data to identify entry and exit points
for (i in 1:(nrow(crypto_log_df) - 1)) {
  if (!in_position && crypto_log_df$Wt[i] <= (mean_wt - delta)) {
    # Enter position: Long LogEthereum, Short LogBitcoin
    crypto_log_df$position[i] <- 1
    in_position <- TRUE
    entry_index <- i
  } else if (in_position && crypto_log_df$Wt[i] >= (mean_wt + delta)) {
    # Exit position
    crypto_log_df$position[i] <- -1
    # Calculate return of the position
    crypto_log_df$return[i] <- crypto_log_df$Wt[i] - crypto_log_df$Wt[entry_index]
    in_position <- FALSE
  } else {
    crypto_log_df$position[i] <- 0
  }
}
crypto_log_df[crypto_log_df$position != 0, ]
crypto_log_df[crypto_log_df$position != 0, ] %>%
  select(Date, LogBitcoin, LogEthereum, Wt,position,return)

# Calculate cumulative returns
crypto_log_df$cumulative_return <- cumsum(crypto_log_df$return)
Total_logReturn = sum(crypto_log_df$return)
```

```
Total_logReturn
trade_df <-crypto_log_df[crypto_log_df$position != 0, ]

# Select required columns and add Bitcoin and Ethereum columns
trade_df <- trade_df %>%
  select(Date, LogBitcoin, LogEthereum, position) %>%
  mutate(
    Bitcoin = exp(LogBitcoin),
    Ethereum = exp(LogEthereum)
  )
trade_df$return <- rep(NA, nrow(trade_df))

# Calculate return for position of -1
for (i in 2:nrow(trade_df)) {
  if (trade_df$position[i] == -1) {
    trade_df$return[i] <- ((1000/trade_df$Bitcoin[i-1])*
    (trade_df$Bitcoin[i]-trade_df$Bitcoin[i-1])) -
      (0.7126339*1000/trade_df$Ethereum[i-1])*
      (trade_df$Ethereum[i]-trade_df$Ethereum[i-1])
  }
}
trade_df

##

library(ggplot2)
library(pracma)

# Generate synthetic data
set.seed(0)
x <- seq(0, 10, length.out = 10)
m <- 1/2
b <- 0
y_true <- m * x + b
y <- y_true + rnorm(length(x), mean = 0, sd = 1)
data <- data.frame(x = x, y = y, y_true = y_true)

# Plot OLS with vertical distances
ggplot(data, aes(x, y)) +
  geom_point(color = "blue") +
```

```
  geom_abline(slope = m, intercept = b, color = "black") + # Longer true line
  geom_segment(aes(xend = x, yend = y_true), color = "red", linetype = "dashed") +
  ggtitle("OLS:␣Vertical␣Distances") +
  coord_fixed(ratio = 1,xlim = c(0, 12), ylim = c(0, 8)) +
  theme_minimal()+
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"), # Add y-axis line
        legend.position = "top")

# True line parameters
c <- y + (x/m)
# Compute intersection points
x_proj <- (c-b) / (m+(1/m))
y_proj <- m * x_proj + b
# Segment data
segments_perp <- data.frame(x = x, y = y, xend = x_proj, yend = y_proj)

# Plot ODR with Perpendicular distances
ggplot(data, aes(x, y)) +
  geom_point(color = "blue") +
  geom_abline(slope = m, intercept = b, color = "black") + # Longer true line
  geom_segment(data = segments_perp, aes(x = x, y = y, xend = xend, yend = yend),
               color = "darkgreen", linetype = "dashed") +
  ggtitle("ODR:␣Perpendicular␣Distances") +
  coord_fixed(ratio = 1,xlim = c(0, 12), ylim = c(0, 8)) +
  theme_minimal()+
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        axis.line.x = element_line(color = "black"), # Add x-axis line
        axis.line.y = element_line(color = "black"), # Add y-axis line
        legend.position = "top")
```

APPENDIX F

SOURCE CODE OF CHAPTER SEVEN

F.1 Univariate GARCH Model

```
library(rugarch)
library(BEKKs)
library(dplyr)
library(zoo)
library(psych)
library(ggplot2)
library(corrplot)
library(tidyr)
library(purrr)
library(stringr)
# time series diagnostics
library(PerformanceAnalytics) # charts and return analytics
library(tseries)           # qqnorm, adf, ljung-box, arch.test (via FinTS)
library(FinTS)             # ArchTest
library(forecast)          # Acf/Pacf plotting niceties

# Join all cryptocurrency dataframes
cryptocurrency_df <- Bitcoin_price %>%
  inner_join(Ethereum_price, by = "Date") %>%
  inner_join(EthereumClassic_price, by = "Date") %>%
  inner_join(TetherUSDt_price, by = "Date") %>%
  inner_join(XRP_price, by = "Date") %>%
  inner_join(Cardano_price, by = "Date") %>%
  inner_join(Dogecoin_price, by = "Date")

# Order the data by date
cryptocurrency_df <- cryptocurrency_df %>%arrange(Date)

# List of columns to compute daily returns for
columns_to_compute <- c("BitcoinClose","EthereumClose", "EthereumClassicClose",
     "TetherUSDtClose", "XRPClose", "CardanoClose", "DogecoinClose")

# Compute daily returns
cryptocurrency_df <- cryptocurrency_df %>%
  mutate(across(all_of(columns_to_compute),
                ~ (.-lag(.))/lag(.),
                .names = "{.col}_return"))
```

```
# Remove rows with any NA values
cryptocurrency_df <- cryptocurrency_df %>% na.omit()

return_df <- cryptocurrency_df %>%
  select(Date, ends_with("_return"))%>% arrange(Date)
head(return_df)

########## Time plot of daily returns
# reset any stale device
graphics.off()
# identify the return columns
return_cols <- names(return_df) %>% setdiff("Date") %>% (\(.) .
[str_ends(., "_return")])()
clean_name <- function(col) {
  col %>%
    str_remove("_return$") %>%
    str_replace("BitcoinClose", "Bitcoin") %>%
    str_replace("EthereumClose", "Ethereum") %>%
    str_replace("EthereumClassicClose", "Ethereum␣Classic") %>%
    str_replace("TetherUSDtClose", "Tether") %>%
    str_replace("XRPClose", "XRP") %>%
    str_replace("CardanoClose", "Cardano") %>%
    str_replace("DogecoinClose", "Dogecoin")
}

# loop and print each series
for (col in return_cols) {
  p <- ggplot(return_df, aes(x = Date, y = .data[[col]])) +
    geom_line(color = "steelblue", linewidth = 0.6) +
    geom_hline(yintercept = 0, color = "grey50", linewidth = 0.4) + # zero line
    labs(
      #title = paste0(col, " daily returns"),
      x = "date", y = paste0(clean_name(col), "␣Daily␣return")
    ) +
    theme_minimal() +
    theme(axis.line.x = element_line(color = "black"), # Add x-axis line
          axis.line.y = element_line(color = "black"), # Add y-axis line
          )
  print(p)
  }
```

```
######### Plot monthly (30-days) volatility ##########
# build an xts object with all return columns (exclude Date)
ret_xts <- xts(return_df %>% select(-Date), order.by = return_df$Date)

# 30-day rolling annualized volatility for each column
realizedvol_xts <- zoo::rollapply(
  ret_xts,
  width = 30, by = 1, align = "right",
  FUN = function(win) {
    apply(win, 2, function(col) PerformanceAnalytics::sd.annualized(col,
    scale = 365))
  },
  fill = NA
)

# convert xts back to a long data.frame (one tidy frame for all cryptos)
vol_df <- data.frame(date = index(realizedvol_xts), coredata(realizedvol_xts),
check.names = FALSE) |>
  pivot_longer(-date, names_to = "series", values_to = "volatility")
head(vol_df)

for (s in unique(vol_df$series)) {
  p <- ggplot(dplyr::filter(vol_df, series == s), aes(x = date, y = volatility)) +
    geom_line(color = "steelblue", linewidth = 0.8) +
    labs(
      x = "date", y = paste0(clean_name(s), "␣volatility␣(annualized)")
    ) +
    theme_minimal()+
    theme(axis.line.x = element_line(color = "black"), # Add x-axis line
          axis.line.y = element_line(color = "black"), # Add y-axis line
    )
  print(p)
}

######### Fit the Garch Model
# Fit a standard sGARCH(1,1) with constant mean and t-distribution
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
  mean.model    = list(armaOrder = c(0,0), include.mean = TRUE),
```

```
  distribution.model = "std"
)

# pick return columns
ret_cols <- grep("_return$", names(return_df), value = TRUE)

# fit each series and print (drop NA/Inf automatically)
for (nm in ret_cols) {
  y <- return_df[[nm]]
  y <- y[is.finite(y)]
  cat("\n==============================\n")
  cat("fitting:", nm, "\n")
  fit_garch <- ugarchfit(spec = garch_spec, data = y)#, solver = "hybrid")
  print(fit_garch)
}
```

F.2 Multivariate GARCH Model

```
########## Fit a BEKK model
# pick the three return columns
keep <- c("BitcoinClose_return", "EthereumClose_return", "EthereumClassicClose_return")

# build an NxK matrix
X <- return_df %>%
  select(all_of(keep)) %>%
  as.matrix()

colnames(X) <- gsub("Close_return", "", colnames(X))
# drop rows with any NA across assets
X <- X[stats::complete.cases(X), ]
X <- scale(X, center = TRUE, scale = FALSE)

# choose model type:
#  - "bekk"  : full BEKK (most flexible, many params)
# add asymmetric = FALSE to use symmetric BEKK , TRUE to allow leverage-type effects,
spec <- bekk_spec(model = list(type = "bekk", asymmetric = FALSE))
fit  <- bekk_fit(spec, X, max_iter = 300, crit = 1e-8)
summary(fit)  # parameter estimates, t-stats, etc.
plot(fit)     # quick vol/cov/corr plots and diagnostics
```

APPENDIX G

SOURCE CODE OF CHAPTER EIGHT

G.1 R Code for the Analysis of Interruptions in Energy Supply

```
### Load data
require(RODBC)
library ('lmtest')
library ('dplyr')
library ('leaps')
library(stats)
library (glmnet)
library(ncvreg)
library(boot)
library(caTools)
library(fastDummies)

conn <- odbcConnect("Databricks_DSN")
df <- sqlQuery(conn, SQL_outage_data)
head(df)
table(df$year,df$weekofyear)
Data <- df[complete.cases(df),]
Data$sinweek<-ifelse(Data$year==2000|Data$year==2001, sin(Data$weekofyear*2*pi/53),
sin(Data$weekofyear*2*pi/52))
Data$sinmonth <-sin(Data$month*2*pi/12)
Data$loutages <- log(Data$daily_outages)

# Create dummy variable
Data <- dummy_cols(Data, select_columns = "storm_type",remove_most_frequent_dummy=TRUE,
 remove_selected_columns=TRUE)
table(df$storm_type)
sum(Data$storm_type_1)
sum(Data$storm_type_2)
sum(Data$storm_type_3)
sum(Data$storm_type_4)
sum(Data$storm_type_5)

###Use independent variables to predict the next day outages######
Data <- Data %>%
  mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
  # Adjust the format to match your date format
  arrange(date) %>%
  mutate(daily_outages = lead(daily_outages),
```

```
        loutages = lead(loutages))
Data <- Data[complete.cases(Data),]
dfcopula <-Data %>% dplyr :: select(-c(loutages,year,month,date,weekofyear,
daily_outagemins,Day,Station,date.1,HEAVY_TSRA,SnowRapidlyIncreasing))

y <- Data %>% dplyr :: select(loutages)%>%as.matrix()
X <- Data %>% dplyr :: select(-c(loutages,year,month,date,weekofyear,daily_outages,
daily_outagemins,Day,Station,date.1,HEAVY_TSRA,SnowRapidlyIncreasing))%>%as.matrix()
X_intercept <-cbind(1,X)
raw_data <- data.frame(y,X)
p <- ncol(X)+1
head(raw_data)

# Histogram and Normality test ##
table(Data$daily_outages)
hist(Data$daily_outages, breaks = 1000,xlim = c(0,50000),main =
'Histogram␣of␣Daily␣Interruptions',xlab = 'Daily␣Interruption')
hist(Data$loutages,breaks = 100, main = 'Histogram␣of␣log␣Daily␣Interruptions',
xlab = 'log␣Daily␣Interruption')

# Perform Shapiro-Wilk Normality test
shapiroResult <- shapiro.test(Data$loutages)
shapiroResult

# Create Q-Q plot
qqPlot <- qqnorm(Data$loutages,pch = 16,cex = 0.3,main = "")
qqline(Data$loutages)

####### Fit Full model and diagnostics plots #####
full_names <- colnames(X)
xcount.FM <- c(full_names)
Formula_FM <- as.formula(paste("loutages␣~",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,raw_data)
summary(model_full)
#write.csv(raw_data,"Research_dataset_model_final.csv")

y2 <- Data %>% dplyr :: select(daily_outages)%>%as.matrix()
raw_data2 <- data.frame(y2,X)
#write.csv(raw_data2,"Research_dataset1005.csv")
Formula_FM2 <- as.formula(paste("daily_outages␣~",paste(xcount.FM, collapse = "+")))
```

```
model_full2 <-lm(Formula_FM2,raw_data2)
summary(model_full2)

# Correlation plot
library(corrplot)
cor_matrix <- cor(Data[,c(80,6,9:20)])
corrplot(cor_matrix, method = "color")

#Regression diagnostics plots
library(ggplot2)
library(broom)
diagnostics <- augment(model_full)

#Residuals vs. Fitted Values:
ggplot(diagnostics, aes(.fitted, .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  xlab("Fitted␣Values") +
  ylab("Residuals") +
  #ggtitle("Residuals vs. Fitted Values") +
  theme_minimal() # Set a white background

#Normal Q-Q Plot:
ggplot(diagnostics, aes(sample = .std.resid)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical␣Quantiles␣of␣Standard␣Normal␣distribution") +
  ylab("Standardized␣Residuals") +
  #ggtitle("Normal Q-Q Plot") +
  theme_minimal() # Set a white background

#Scale-Location Plot:
ggplot(diagnostics, aes(.fitted, sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_smooth() +
  xlab("Fitted␣Values") +
  ylab("Square␣Root␣of␣Standardized␣Residuals") +
  #ggtitle("Scale-Location Plot")
  theme_minimal() # Set a white background
```

```
#Scale-Location Plot:
ggplot(diagnostics, aes(.fitted, sqrt(abs(.std.resid)))) +
  geom_point() +
  #geom_smooth() +
  xlab("Fitted␣Values") +
  ylab("Square␣Root␣of␣Standardized␣Residuals") +
  #ggtitle("Scale-Location Plot")
  theme_minimal() # Set a white background

#Residuals vs. Leverage plot
library(car)
leverage <- hatvalues(model_full)
residuals <- rstandard(model_full)
data <- data.frame(Leverage = leverage, Residuals = residuals)
ggplot(data, aes(Leverage, Residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  geom_smooth(se = FALSE, color = "blue") +
  xlab("Leverage") +
  ylab("Standardized␣Residuals") +
  #ggtitle("Residuals vs. Leverage Plot")
  theme_minimal() # Set a white background

#Defining Shrinkage and Positive Shrinkage estimation functions
Shrinkage_Est <- function(beta_FM,beta_SM,test_stat,p2){
  return (beta_FM - ((beta_FM - beta_SM)*(p2-2)/test_stat))
}
PShrinkage_Est <-function(beta_FM,beta_SM,test_stat ,p2){
  return (beta_SM + max(0,(1-(p2-2)/test_stat))*(beta_FM - beta_SM))
}
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}

########## Submodel estimation #########

#Submodel by BIC
model_full <- lm(Formula_FM, data = raw_data)
step_model <- step(model_full, direction = "both",k = log(nrow(raw_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
full_names <- colnames(X)
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
```

```
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
Formula_SM.BIC <- as.formula(paste("loutages␣~",paste(xcount.SM,
collapse = "+")))
summary(step_model)

#Submodel by AIC
model_full <- lm(Formula_FM, data = raw_data)
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste("loutages␣~",paste(xcount.SM.AIC,
collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y , alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
p2.lasso <- p-p1.lasso
#Formula of the Sub model
xcount.SM.lasso <- c(sub_names.lasso)
Formula_SM.lasso <- as.formula(paste("loutages␣~",paste(xcount.SM.lasso,
collapse = "+")))
submodel_lasso <-lm(Formula_SM.lasso,raw_data)
summary(submodel_lasso)

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
```

```
B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)
PE100_values_bootstrap <- matrix(0, nrow = B, ncol = 15)

# Set the seed for reproducibility
set.seed(1234)

for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(raw_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- raw_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(loutages) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-loutages) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(loutages) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-loutages) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
  df_train <- data.frame( y_train, X_train)
  #The full model fit based on train data
  fit_FM <-lm(Formula_FM,df_train)
  beta.FM <- fit_FM$coef
  beta.FM[is.na(beta.FM)] <- 0
  #The sub model fit based on train data
  fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
  beta.SM.BIC <- rep(0,p)
  beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
  beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
  fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
```

```
beta.SM.AIC <- rep(0,p)
beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
beta.SM.lasso <- rep(0,p)
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE, standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)
#the penalty parameter is tuned to obtain optimal prediction
#(minimize out of sample error via grid-search cross validation)
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
```

```
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge
yhat_df <- data.frame(y_test_scale,yhat_beta.FM,yhat_beta.SM.BIC,
    yhat_beta.S.BIC ,
    yhat_beta.PS.BIC ,
    yhat_beta.SM.AIC ,
    yhat_beta.S.AIC ,
    yhat_beta.PS.AIC ,
    yhat_beta.SM.lasso ,
    yhat_beta.S.lasso ,
    yhat_beta.PS.lasso ,
    yhat_beta.lasso ,
    yhat_beta.enet ,
    yhat_beta.alasso ,
```

```
    yhat_beta.scad ,
    yhat_beta.ridge )
#Calculate prediction errors of estimators
PE_values <- c(FM=Prediction_Error(y_test_scale, yhat_beta.FM),
               SM.BIC=Prediction_Error(y_test_scale, yhat_beta.SM.BIC),
               S.BIC =Prediction_Error(y_test_scale, yhat_beta.S.BIC),
               PS.BIC=Prediction_Error(y_test_scale, yhat_beta.PS.BIC),
               SM.AIC=Prediction_Error(y_test_scale, yhat_beta.SM.AIC),
               S.AIC =Prediction_Error(y_test_scale, yhat_beta.S.AIC),
               PS.AIC=Prediction_Error(y_test_scale, yhat_beta.PS.AIC),
               SM.lasso=Prediction_Error(y_test_scale, yhat_beta.SM.lasso),
               S.lasso =Prediction_Error(y_test_scale, yhat_beta.S.lasso),
               PS.lasso=Prediction_Error(y_test_scale, yhat_beta.PS.lasso),
               LASSO = Prediction_Error( y_test_scale, yhat_beta.lasso),
               ENET = Prediction_Error( y_test_scale, yhat_beta.enet),
               ALASSO = Prediction_Error( y_test_scale, yhat_beta.alasso),
               SCAD = Prediction_Error( y_test_scale , yhat_beta.scad ),
               RIDGE = Prediction_Error( y_test_scale , yhat_beta.ridge)
               )
PE_values_bootstrap[i,] <- PE_values
#Calculate prediction errors of estimators for top 100 y values.
FM_100 <- head(cbind(y_test_scale, yhat_beta.FM)[order(y_test_scale,
decreasing = TRUE), ], 100)
SM.BIC_100 <- head(cbind(y_test_scale, yhat_beta.SM.BIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
S.BIC_100 <- head(cbind(y_test_scale, yhat_beta.S.BIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
PS.BIC_100 <- head(cbind(y_test_scale, yhat_beta.PS.BIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
SM.AIC_100 <- head(cbind(y_test_scale, yhat_beta.SM.AIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
S.AIC_100 <- head(cbind(y_test_scale, yhat_beta.S.AIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
PS.AIC_100 <- head(cbind(y_test_scale, yhat_beta.PS.AIC)[order(y_test_scale,
decreasing = TRUE), ], 100)
SM.lasso_100 <- head(cbind(y_test_scale, yhat_beta.SM.lasso)[order(y_test_scale,
decreasing = TRUE), ], 100)
S.lasso_100 <- head(cbind(y_test_scale, yhat_beta.S.lasso)[order(y_test_scale,
decreasing = TRUE), ], 100)
PS.lasso_100 <- head(cbind(y_test_scale, yhat_beta.PS.lasso)[order(y_test_scale,
```

```
  decreasing = TRUE), ], 100)
  lasso_100 <- head(cbind(y_test_scale, yhat_beta.lasso)[order(y_test_scale,
  decreasing = TRUE), ], 100)
  enet_100 <- head(cbind(y_test_scale, yhat_beta.enet)[order(y_test_scale,
  decreasing = TRUE), ], 100)
  alasso_100 <- head(cbind(y_test_scale, yhat_beta.alasso)[order(y_test_scale,
  decreasing = TRUE), ], 100)
  scad_100 <- head(cbind(y_test_scale, yhat_beta.scad)[order(y_test_scale,
  decreasing = TRUE), ], 100)
  ridge_100 <- head(cbind(y_test_scale, yhat_beta.ridge)[order(y_test_scale,
  decreasing = TRUE), ], 100)
  PE100_values <- c(FM=Prediction_Error(FM_100[,1], FM_100[,2]),
  SM.BIC = Prediction_Error(SM.BIC_100[, 1], SM.BIC_100[, 2]),
  S.BIC = Prediction_Error(S.BIC_100[, 1], S.BIC_100[, 2]),
  PS.BIC = Prediction_Error(PS.BIC_100[, 1], PS.BIC_100[, 2]),
  SM.AIC = Prediction_Error(SM.AIC_100[, 1], SM.AIC_100[, 2]),
  S.AIC = Prediction_Error(S.AIC_100[, 1], S.AIC_100[, 2]),
  PS.AIC = Prediction_Error(PS.AIC_100[, 1], PS.AIC_100[, 2]),
  SM.lasso = Prediction_Error(SM.lasso_100[, 1], SM.lasso_100[, 2]),
  S.lasso = Prediction_Error(S.lasso_100[, 1], S.lasso_100[, 2]),
  PS.lasso = Prediction_Error(PS.lasso_100[, 1], PS.lasso_100[, 2]),
  lasso = Prediction_Error(lasso_100[, 1], lasso_100[, 2]),
  enet = Prediction_Error(enet_100[, 1], enet_100[, 2]),
  alasso = Prediction_Error(alasso_100[, 1], alasso_100[, 2]),
  scad = Prediction_Error(scad_100[, 1], scad_100[, 2]),
  ridge = Prediction_Error(ridge_100[, 1], ridge_100[, 2]) )
  PE100_values_bootstrap[i,] <- PE100_values
}

PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))

# Add row names
row_names <- c("FW","SM.BIC","S.BIC","PS.BIC","SM.AIC","S.AIC","PS.AIC","SM.lasso",
"S.lasso","PS.lasso", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
PE.result

PE100_avg <- apply(PE100_values_bootstrap,2,mean)
```

```
PE100_std <- apply(PE100_values_bootstrap,2,sd)
PE100.result <- as.data.frame(cbind(PE100_avg, PE100_std))

# Add row names
row_names <- c("FW","SM.BIC","S.BIC","PS.BIC","SM.AIC","S.AIC","PS.AIC","SM.lasso",
"S.lasso","PS.lasso", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE100.result) <- row_names
PE100.result

####Analysis of the Entire data##########
#The full model fit based on entire data
fit_FM <-lm(Formula_FM,raw_data)
beta.FM <- fit_FM$coef
beta.FM[is.na(beta.FM)] <- 0
summary(fit_FM)

#The sub model fit based on entire data
fit_SM.BIC <-lm(Formula_SM.BIC, raw_data)
beta.SM.BIC <- rep(0,p)
beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
summary(fit_SM.BIC)

fit_SM.AIC <-lm(Formula_SM.AIC, raw_data)
beta.SM.AIC <- rep(0,p)
beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
beta.SM.AIC[is.na(beta.SM.AIC)] <- 0

fit_SM.lasso <-lm(Formula_SM.lasso, raw_data)
beta.SM.lasso <- rep(0,p)
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef

lasso.fit <- glmnet(X, y, alpha = 1,intercept = TRUE, standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]

#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
```

```
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X,y,alpha = alphas[ind],intercept=T,standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]

#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X, y, alpha = 1, penalty.factor = 1/abs(weight),intercept = T,
standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]

scad.fit <- ncvreg(X, y, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
b.scad <- coef(scad.fit, lambda = lam)

ridge.fit <- glmnet( X, y, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]

#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]

beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
```

```
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)

#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_intercept %*% beta.FM
yhat_beta.SM.BIC <- X_intercept %*% beta.SM.BIC
yhat_beta.S.BIC <- X_intercept %*% beta.S.BIC
yhat_beta.PS.BIC <- X_intercept %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_intercept %*% beta.SM.AIC
yhat_beta.S.AIC <- X_intercept %*% beta.S.AIC
yhat_beta.PS.AIC <- X_intercept %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_intercept %*% beta.SM.lasso
yhat_beta.S.lasso <- X_intercept %*% beta.S.lasso
yhat_beta.PS.lasso <- X_intercept %*% beta.PS.lasso
yhat_beta.lasso <- X_intercept %*%b.lasso
yhat_beta.enet <- X_intercept %*%b.enet
yhat_beta.alasso <- X_intercept %*%b.alasso
yhat_beta.scad <- X_intercept %*%b.scad
yhat_beta.ridge <- X_intercept %*%b.ridge

#Calculate prediction errors of estimators
PE_values <- c(FM=Prediction_Error(y, yhat_beta.FM),
               SM.BIC=Prediction_Error(y, yhat_beta.SM.BIC),
               S.BIC =Prediction_Error(y, yhat_beta.S.BIC),
               PS.BIC=Prediction_Error(y, yhat_beta.PS.BIC),
               SM.AIC=Prediction_Error(y, yhat_beta.SM.AIC),
               S.AIC =Prediction_Error(y, yhat_beta.S.AIC),
               PS.AIC=Prediction_Error(y, yhat_beta.PS.AIC),
               SM.lasso=Prediction_Error(y, yhat_beta.SM.lasso),
               S.lasso =Prediction_Error(y, yhat_beta.S.lasso),
               PS.lasso=Prediction_Error(y, yhat_beta.PS.lasso),
               LASSO = Prediction_Error( y, yhat_beta.lasso),
               ENET = Prediction_Error( y, yhat_beta.enet),
               ALASSO = Prediction_Error( y, yhat_beta.alasso),
               SCAD = Prediction_Error( y , yhat_beta.scad ),
               RIDGE = Prediction_Error( y , yhat_beta.ridge)
)
PE_values
n=nrow(y)
n=1000
```

```
#Calculate prediction errors of estimators for top 100 y values.
FM_100 <- head(cbind(y, yhat_beta.FM)[order(y, decreasing = TRUE), ], n)
SM.BIC_100 <- head(cbind(y, yhat_beta.SM.BIC)[order(y, decreasing = TRUE), ], n)
S.BIC_100 <- head(cbind(y, yhat_beta.S.BIC)[order(y, decreasing = TRUE), ], n)
PS.BIC_100 <- head(cbind(y, yhat_beta.PS.BIC)[order(y, decreasing = TRUE), ], n)
SM.AIC_100 <- head(cbind(y, yhat_beta.SM.AIC)[order(y, decreasing = TRUE), ], n)
S.AIC_100 <- head(cbind(y, yhat_beta.S.AIC)[order(y, decreasing = TRUE), ], n)
PS.AIC_100 <- head(cbind(y, yhat_beta.PS.AIC)[order(y, decreasing = TRUE), ], n)
SM.lasso_100 <- head(cbind(y, yhat_beta.SM.lasso)[order(y, decreasing = TRUE), ], n)
S.lasso_100 <- head(cbind(y, yhat_beta.S.lasso)[order(y, decreasing = TRUE), ], n)
PS.lasso_100 <- head(cbind(y, yhat_beta.PS.lasso)[order(y, decreasing = TRUE), ], n)
lasso_100 <- head(cbind(y, yhat_beta.lasso)[order(y, decreasing = TRUE), ], n)
enet_100 <- head(cbind(y, yhat_beta.enet)[order(y, decreasing = TRUE), ], n)
alasso_100 <- head(cbind(y, yhat_beta.alasso)[order(y, decreasing = TRUE), ], n)
scad_100 <- head(cbind(y, yhat_beta.scad)[order(y, decreasing = TRUE), ], n)
ridge_100 <- head(cbind(y, yhat_beta.ridge)[order(y, decreasing = TRUE), ], n)

PE100_values <- c(FM=Prediction_Error(FM_100[,1], FM_100[,2]),
SM.BIC = Prediction_Error(SM.BIC_100[, 1], SM.BIC_100[, 2]),
S.BIC = Prediction_Error(S.BIC_100[, 1], S.BIC_100[, 2]),
PS.BIC = Prediction_Error(PS.BIC_100[, 1], PS.BIC_100[, 2]),
SM.AIC = Prediction_Error(SM.AIC_100[, 1], SM.AIC_100[, 2]),
S.AIC = Prediction_Error(S.AIC_100[, 1], S.AIC_100[, 2]),
PS.AIC = Prediction_Error(PS.AIC_100[, 1], PS.AIC_100[, 2]),
SM.lasso = Prediction_Error(SM.lasso_100[, 1], SM.lasso_100[, 2]),
S.lasso = Prediction_Error(S.lasso_100[, 1], S.lasso_100[, 2]),
PS.lasso = Prediction_Error(PS.lasso_100[, 1], PS.lasso_100[, 2]),
lasso = Prediction_Error(lasso_100[, 1], lasso_100[, 2]),
enet = Prediction_Error(enet_100[, 1], enet_100[, 2]),
alasso = Prediction_Error(alasso_100[, 1], alasso_100[, 2]),
scad = Prediction_Error(scad_100[, 1], scad_100[, 2]),
ridge = Prediction_Error(ridge_100[, 1], ridge_100[, 2]) )
PE100_values
```

G.2 SAS Code and R Code for Copula Transformation Based Model

```
title "Copula Transformation for Skewed Dataset";
/* Transform skewed multiple variate dataset */
/* Import dataset from csv file, which is exported from SQL query and R pre-process */
```

```
FILENAME REFFILE '/home/minjieyu1/Reseach for Copula/Research_dataset.csv';
PROC IMPORT DATAFILE=REFFILE OUT=importData DBMS=CSV;
  GETNAMES=YES;
RUN;

proc print data=importData(obs=15);
run;
PROC CONTENTS DATA=importData;
RUN;

%let x1= MaxTemp MinTemp MaxDew MinDew Precip AvWindSpeed AvWindDrct MinRH AvRH MaxRH
SnowFall SnowDepth MaxWindSpeed MaxWindGust programA_past3m programA_past6m
programA_past1y programA_past2y programA_past3y programB_past3m programB_past6m
programB_past1y programB_past2y programB_past3y programC_past3m programC_past6m
programC_past1y programC_past2y programC_past3y programD_past3m programD_past6m
programD_past1y programD_past2y programD_past3y programE_past3m programE_past6m
programE_past1y programE_past2y programE_past3y programF_past3m programF_past6m
programF_past1y programF_past2y programF_past3y sinweek sinmonth ;

%let x2 = Lightning FREEZING_RAIN MODERATE_RAIN HEAVY_RAIN LIGHT_RAIN
MODERATE_Thunderstorm
HEAVY_Thunderstorm LIGHT_Thunderstorm MODERATE_TSRA LIGHT_TSRA FREEZING_DRIZZLE
MODERATE_DRIZZLE HEAVY_DRIZZLE LIGHT_DRIZZLE SQUALLS HEAVY_HAIL LIGHT_HAIL
LightningCloudToCloud LightningCloudToAir LightningCloudToGround IcePellets
SnowPellets storm_type_1 storm_type_2 storm_type_3 storm_type_4 storm_type_5;

%let y=daily_outages;

data importData;
  set importData;
  /*store the observation number (n)*/
  sr=_n_;
  raw_y = daily_outages;
run;

/*running regression on raw data*/;
proc reg data=importData;
  model &y = &x1 &x2 /rsquare p adjrsq;
run;
```

```
title"Copula transformation for dataset";
/*simulating (mult.)normal distribution for numerical indep vars and response*/;
proc copula data=importData;
  var &y &x1;
  fit normal /marginals=empirical
  outcopula=outcopula_correlation
  outpseudo=outpseudo_uniform; *pseudo-samples with uniform marginal distributions;
  simulate /ndraws=2000
    seed=123456
    out=returns2_normal outuniform=outuniform2_normal;
run;

proc print data=outcopula_correlation;
  title2 "outcopula_correlation correlation estimates";
run;

proc print data=outpseudo_uniform(obs=15);
  title2 "pseudo-samples with uniform marginal distributions";
run;
data transf_normal;
  set outpseudo_uniform;
  /*store the observation number (n)*/
  sr=_n_;
  /*store the normal quantile of the variable y*/
  ystar=quantile("Normal", daily_outages);
  /*define two arrays ww and xx, both with &k elements*/
  array ww [46] w1-w46;
  /* Using array equations for repetitive equations*/;
  array xx [46] &x1;
  do i=1 to 46;
    ww[i]=quantile("Normal", xx[i]);
    *normal quantile of xi;
  end;
  drop i;
  /* drop the temporary variable "i" from the output dataset. */
run;

proc print data=transf_normal(obs=15);
  title2 "transform normal";
run;
```

```
/*add dummy and binary variables back to the dataset*/
data copula_trans_data;
    merge transf_normal importData;
    by sr;
    keep sr raw_y ystar w1-w46 &x2; /* Specify the columns you want to keep */
run;

proc print data=copula_trans_data(obs=10);
  title2 "Copula transformed data and Original binary variables and response";
run;

PROC EXPORT DATA=copula_trans_data
    OUTFILE='/home/minjieyu1/Reseach for Copula/copula_trans_data_nextday.csv'
    DBMS=CSV REPLACE;
RUN;

/*regressing ystar on w1-wk and original binary variables*/;
proc reg data=copula_trans_data outest=copest;
model ystar= w1-w46 &x2/rsquare p adjrsq;*running reg on copula data;
id sr;
output out=myoutput p=pyhat
lcl=lcly ucl=ucly lclm=lcmy uclm=ucmy R=N_copularesidual;
/*predicted values (pyhat), lower and upper confidence limits for the predicted values
(lcly, ucly), lower and upper confidence limits for the mean response (lcmy, ucmy)*/
run;

proc print data=myoutput(obs=15);
title"regression output for normalised dep variables";
run;
proc print data=copest;title"copest";
run;

data cdf_normal;
set myoutput;
/*generating uniform for ystar,pyhat,lcly,ucly,lcmy and ucmy by using cdf*/;
uystar=cdf('normal',ystar);
upyhat=cdf('normal',pyhat);
ulcly=cdf('normal',lcly);
uucly=cdf('normal',ucly);
```

```
ulcmy=cdf('normal',lcmy);
uucmy=cdf('normal',ucmy);
run;

proc print data=cdf_normal(obs=15);
title"uniform distribution from cdf function";
var ystar pyhat uystar upyhat sr;
run;

proc sort data=cdf_normal;
by sr;
run;

proc print data=returns2_normal(obs=15);
title"returns2_normal";
run;

proc print data=outuniform2_normal(obs=15);
title"outuniform2_normal";
run;

data outuniform2_normal;
set outuniform2_normal;
uy=&y;
array uxx [46] ux1-ux46 ;
/*Using array equations for repetitive equations*/;
array xx [46] &x1 ;
do i = 1 to 46;
uxx[i] = xx[i];
end;
drop i;
keep uy ux1-ux73;
run;
proc print data=outuniform2_normal(obs=15);run;

data both;
merge returns2_normal outuniform2_normal;
/*returns2_normal is simulating (for this example 1000) observations for the given
data set, outuniform2_normal is converting the (for this example 1000) simulated
observations from returns2_normal data set into uniform distribution*/;
```

```
/* Merging data set from simulated output with copula output*/;
run;

proc sort data=both;
by uy;
run;

proc print data=both(obs=15);
title "Both (simulated samples and simulated uniform from copula output)";
run;

data export_simulation_y;
set both;
keep uy &y;
run;

PROC EXPORT DATA=export_simulation_y
    OUTFILE='/home/minjieyu1/Reseach for Copula/export_simulation_y_nextday.csv'
    DBMS=CSV REPLACE;
RUN;
```

R code of the Copula Transformation Based Model for the Energy supply data.

```
library ('lmtest')
library ('dplyr')
library ('leaps')
library(stats)
library (glmnet)
library(ncvreg)
library(boot)
library(caTools)
library(fastDummies)
library(tidyr)

#Defining Shrinkage and Positive Shrinkage estimation functions
Shrinkage_Est <- function(beta_FM,beta_SM,test_stat,p2){
  return (beta_FM - ((beta_FM - beta_SM)*(p2-2)/test_stat))
}
PShrinkage_Est <-function(beta_FM,beta_SM,test_stat ,p2){
  return (beta_SM + max(0,(1-(p2-2)/test_stat))*(beta_FM - beta_SM))
```

```
}
Prediction_Error <- function(y,yhat){mean((y-yhat)^2)}

#load copula transformed data from SAS, with original response variable that
#will be used to compute prediction error, observation number.
setwd("C:/Users/Desktop/Research")
simulationData <- read.csv("export_simulation_y.csv")
data <- read.csv("copula_trans_data.csv")
#For next day prediction model
simulationData <- read.csv("export_simulation_y_nextday.csv")
data <- read.csv("copula_trans_data_nextday.csv")
head(data)
head(simulationData)

#define response and independent variables
y <- data %>% dplyr :: select(ystar,sr,raw_y)%>%as.matrix()
X <- data %>% dplyr :: select(-c(ystar,sr,raw_y))%>%as.matrix()
model_data <- data.frame(y,X)
p <- ncol(X)+1
X_intercept <-cbind(1,X)

#fit the full model
full_names <- colnames(X)
xcount.FM <- c(full_names)
Formula_FM <- as.formula(paste("ystar␣~",paste(xcount.FM, collapse = "+")))
model_full <-lm(Formula_FM,model_data)
summary(model_full)

######### Submodel estimation #########
#Submodel by BIC
step_model <- step(model_full, direction = "both",k = log(nrow(model_data)))
sub_names.BIC <- names(step_model$coefficients)[-1]
p1_indx.BIC <- c(1,which(full_names %in% sub_names.BIC)+1)
p1.BIC <- length(p1_indx.BIC)+1 #the value of p1
p2.BIC <- p-p1.BIC #the value of p2
#Formula of the Sub model
xcount.SM <- c(sub_names.BIC)
Formula_SM.BIC <- as.formula(paste("ystar␣~",paste(xcount.SM, collapse = "+")))
summary(step_model)
```

```
#Submodel by AIC
step_model <- step(model_full, direction = "both")
sub_names.AIC <- names(step_model$coefficients)[-1]
p1_indx.AIC <- c(1,which(full_names %in% sub_names.AIC)+1)
p1.AIC <- length(p1_indx.AIC)+1 #the value of p1
p2.AIC <- p-p1.AIC #the value of p2
xcount.SM.AIC <- c(sub_names.AIC)
Formula_SM.AIC <- as.formula(paste("ystar ~",paste(xcount.SM.AIC, collapse = "+")))
summary(step_model)

#Submodel by LASSO
lasso.fit <- glmnet(X, y[, "ystar"], alpha = 1,intercept = F, standardize = F)
lasso.bic = deviance(lasso.fit) + log(NROW(X))*lasso.fit$df
b.lasso = coef(lasso.fit)[-1, which.min(lasso.bic)]
sub_names.lasso <- names(b.lasso[b.lasso !=0])
p1_indx.lasso <- c(1,which(full_names %in% sub_names.lasso)+1)
p1.lasso <- length(p1_indx.lasso)+1 #the value of p1
p2.lasso <- p-p1.lasso
#Formula of the Sub model
xcount.SM.lasso <- c(sub_names.lasso)
Formula_SM.lasso <- as.formula(paste("ystar ~",paste(xcount.SM.lasso, collapse = "+")))

# Create empty lists to store bootstrap samples
bootstrap_samples <- list()
train_sets <- list()
test_sets <- list()
simulationData$t <- simulationData$uy
simulationData$y <- simulationData$daily_outages

#create a function to replace Missing prediction values in original scale by mean of lag
#and lead values from simulation;
update_pred <- function(resultdf,uyhat){
  tempdf <- resultdf
  tempdf$t <-tempdf[[uyhat]]
  df <- bind_rows(simulationData, tempdf) %>%
    select(y, t, raw_y) %>%
    arrange(t) %>%
    mutate(ylead = coalesce(lead(y), lead(y, 2), lead(y, 3), lead(y, 4), lead(y, 5)),
           ylag = coalesce(lag(y), lag(y, 2), lag(y, 3), lag(y, 4), lag(y, 5)),
           y = ifelse(is.na(y), (ylead+ylag)/2, y))
```

```
  df <- df[complete.cases(df$raw_y), ] # Remove rows with NA values in raw_y
  df <- df[complete.cases(df$y), ]
  return(df)
}

B <- 250
PE_values_bootstrap <- matrix(0, nrow = B, ncol = 15)
PE100_values_bootstrap <- matrix(0, nrow = B, ncol = 15)
# Set the seed for reproducibility
set.seed(1234)

for (i in 1:B) {
  # Generate bootstrap sample indices
  bootstrap_indices <- sample(nrow(model_data), replace = TRUE)
  # Create bootstrap sample
  bootstrap_sample <- model_data[bootstrap_indices, ]
  # Store bootstrap sample
  bootstrap_samples[[i]] <- bootstrap_sample
  # Split bootstrap sample into train and test sets
  set.seed(i) # Set a different seed for each iteration
  split <- sample.split(bootstrap_sample, SplitRatio = 0.75)
  train_set <- subset(bootstrap_sample, split == TRUE)
  test_set <- subset(bootstrap_sample, split == FALSE)
  #For y
  y_train <- train_set %>% dplyr :: select(ystar) %>% as.matrix()
  #For X
  X_train <- train_set %>% dplyr :: select(-c(ystar,sr,raw_y)) %>% as.matrix()
  #in the test set
  y_test <- test_set %>% dplyr :: select(c(ystar,raw_y)) %>% as.matrix()
  X_test <- test_set %>% dplyr :: select(-c(ystar,sr,raw_y)) %>% as.matrix()
  y_test_scale <- y_test
  X_test_scale <-cbind(1,X_test)
  #dataframe based on train data
  df_train <- data.frame( y_train, X_train)
  #The full model fit based on train data
  fit_FM <-lm(Formula_FM,df_train)
  #summary(fit_FM)
  beta.FM <- fit_FM$coef
  beta.FM[is.na(beta.FM)] <- 0
  #The sub model fit based on train data
```

```
fit_SM.BIC <-lm(Formula_SM.BIC, df_train)
beta.SM.BIC <- rep(0,p)
beta.SM.BIC[p1_indx.BIC] <- fit_SM.BIC$coef
beta.SM.BIC[is.na(beta.SM.BIC)] <- 0
fit_SM.AIC <-lm(Formula_SM.AIC, df_train)
beta.SM.AIC <- rep(0,p)
beta.SM.AIC[p1_indx.AIC] <- fit_SM.AIC$coef
beta.SM.AIC[is.na(beta.SM.AIC)] <- 0
fit_SM.lasso <-lm(Formula_SM.lasso, df_train)
beta.SM.lasso <- rep(0,p)
beta.SM.lasso[p1_indx.lasso] <- fit_SM.lasso$coef
beta.SM.lasso[is.na(beta.SM.lasso)] <- 0
lasso.fit <- glmnet(X_train, y_train, alpha = 1,intercept = TRUE,
standardize = TRUE)
lasso.bic = deviance(lasso.fit) + log(NROW(X_train))*lasso.fit$df
b.lasso = coef(lasso.fit)[,which.min(lasso.bic)]
#ENET
alphas <- seq(0.1,0.9, length.out = 9)
#ind<- 1
fits.enet<-list()
for (ind in 1:length(alphas)){
  fits.enet[[ind]]<- glmnet(X_train,y_train,alpha = alphas[ind],intercept=T,
  standardize = T)}
cvs.enet <- sapply(fits.enet,function(x){
  BIC = deviance(x)+log(NROW(X_train))*x$df
  beta.enet = coef(x)[,which.min(BIC)]
  lamda = alphas
  return(list(PE= mean((y[, "ystar"] - X_intercept%*%beta.enet)^2),
              beta.enet = beta.enet))})
ind_opt <-which.min(do.call(rbind,cvs.enet[1,]))
b.enet <-do.call(rbind,cvs.enet[2,])[ind_opt,]
#ALASSO based on BIC
weight <- b.lasso[-1]
weight <- ifelse(weight == 0, 0.00001, weight)
alasso.fit <-glmnet(X_train, y_train, alpha = 1, penalty.factor = 1/abs(weight),
intercept = T, standardize = T)
alasso.bic = deviance(alasso.fit)+log(NROW(X_train))*alasso.fit$df
b.alasso = coef(alasso.fit)[, which.min(alasso.bic)]
scad.fit <- ncvreg(X_train, y_train, penalty =c("SCAD"))
lam <- scad.fit$lambda[which.min(BIC(scad.fit))]
```

```
b.scad <- coef(scad.fit, lambda = lam)
ridge.fit <- glmnet( X_train, y_train, alpha = 0,intercept = T, standardize = T)
ridge.bic = deviance(ridge.fit)+log(NROW(X_train))*ridge.fit$df
b.ridge = coef(ridge.fit)[, which.min(ridge.bic)]
#Likelihood ratio test
test_LR.BIC <- lrtest(fit_SM.BIC, fit_FM)
test_stat.BIC <- test_LR.BIC$Chisq[2]
test_LR.AIC <- lrtest(fit_SM.AIC, fit_FM)
test_stat.AIC <- test_LR.AIC$Chisq[2]
test_LR.lasso <- lrtest(fit_SM.lasso, fit_FM)
test_stat.lasso <- test_LR.lasso$Chisq[2]
beta.S.BIC <- Shrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.PS.BIC <- PShrinkage_Est(beta.FM, beta.SM.BIC, test_stat.BIC,p2.BIC)
beta.S.AIC <- Shrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.PS.AIC <- PShrinkage_Est(beta.FM, beta.SM.AIC, test_stat.AIC,p2.AIC)
beta.S.lasso <- Shrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
beta.PS.lasso <- PShrinkage_Est(beta.FM, beta.SM.lasso, test_stat.lasso,p2.lasso)
#Estimate Prediction Errors based on test data
yhat_beta.FM <- X_test_scale %*% beta.FM
yhat_beta.SM.BIC <- X_test_scale %*% beta.SM.BIC
yhat_beta.S.BIC <- X_test_scale %*% beta.S.BIC
yhat_beta.PS.BIC <- X_test_scale %*% beta.PS.BIC
yhat_beta.SM.AIC <- X_test_scale %*% beta.SM.AIC
yhat_beta.S.AIC <- X_test_scale %*% beta.S.AIC
yhat_beta.PS.AIC <- X_test_scale %*% beta.PS.AIC
yhat_beta.SM.lasso <- X_test_scale %*% beta.SM.lasso
yhat_beta.S.lasso <- X_test_scale %*% beta.S.lasso
yhat_beta.PS.lasso <- X_test_scale %*% beta.PS.lasso
yhat_beta.lasso <- X_test_scale %*%b.lasso
yhat_beta.enet <- X_test_scale %*%b.enet
yhat_beta.alasso <- X_test_scale %*%b.alasso
yhat_beta.scad <- X_test_scale %*%b.scad
yhat_beta.ridge <- X_test_scale %*%b.ridge

#Transform yhat to original scale, using Copula simulation dataset
# Generating uniform variables using CDF
uyhat_beta.FM <- pnorm(yhat_beta.FM )
uyhat_beta.SM.BIC <- pnorm(yhat_beta.SM.BIC )
uyhat_beta.S.BIC <- pnorm(yhat_beta.S.BIC )
uyhat_beta.PS.BIC <- pnorm(yhat_beta.PS.BIC )
```

```
uyhat_beta.SM.AIC <- pnorm(yhat_beta.SM.AIC )
uyhat_beta.S.AIC <- pnorm(yhat_beta.S.AIC )
uyhat_beta.PS.AIC <- pnorm(yhat_beta.PS.AIC )
uyhat_beta.SM.lasso <- pnorm(yhat_beta.SM.lasso )
uyhat_beta.S.lasso <- pnorm(yhat_beta.S.lasso )
uyhat_beta.PS.lasso <- pnorm(yhat_beta.PS.lasso )
uyhat_beta.lasso <- pnorm(yhat_beta.lasso )
uyhat_beta.enet <- pnorm(yhat_beta.enet )
uyhat_beta.alasso <- pnorm(yhat_beta.alasso )
uyhat_beta.scad <- pnorm(yhat_beta.scad )
uyhat_beta.ridge <- pnorm(yhat_beta.ridge )
resultdf <- data.frame(y_test_scale,uyhat_beta.FM,uyhat_beta.SM.BIC,
uyhat_beta.S.BIC ,uyhat_beta.PS.BIC ,uyhat_beta.SM.AIC ,uyhat_beta.S.AIC,
uyhat_beta.PS.AIC ,uyhat_beta.SM.lasso ,uyhat_beta.S.lasso ,uyhat_beta.PS.lasso,
uyhat_beta.lasso ,uyhat_beta.enet,uyhat_beta.alasso ,uyhat_beta.scad ,uyhat_beta.ridge)
FMdf<- update_pred(resultdf, "uyhat_beta.FM")
SM.BICdf<- update_pred(resultdf, "uyhat_beta.SM.BIC")
S.BICdf<- update_pred(resultdf, "uyhat_beta.S.BIC")
PS.BICdf<- update_pred(resultdf, "uyhat_beta.PS.BIC")
SM.AICdf<- update_pred(resultdf, "uyhat_beta.SM.AIC")
S.AICdf<- update_pred(resultdf, "uyhat_beta.S.AIC")
PS.AICdf<- update_pred(resultdf, "uyhat_beta.PS.AIC")
SM.lassodf<- update_pred(resultdf, "uyhat_beta.SM.lasso")
S.lassodf<- update_pred(resultdf, "uyhat_beta.S.lasso")
PS.lassodf<- update_pred(resultdf, "uyhat_beta.PS.lasso")
lassodf<- update_pred(resultdf, "uyhat_beta.lasso")
enetdf<- update_pred(resultdf, "uyhat_beta.enet")
alassodf<- update_pred(resultdf, "uyhat_beta.alasso")
scaddf<- update_pred(resultdf, "uyhat_beta.scad")
ridgedf<- update_pred(resultdf, "uyhat_beta.ridge")
#Calculate prediction errors of estimators
PE_star <- c(FM=Prediction_Error(log(FMdf$raw_y),log(FMdf$y)),
             SM.BIC=Prediction_Error(log(SM.BICdf$raw_y),log(SM.BICdf$y)),
             S.BIC=Prediction_Error(log(S.BICdf$raw_y),log(S.BICdf$y)),
             PS.BIC=Prediction_Error(log(PS.BICdf$raw_y),log(PS.BICdf$y)),
             SM.AIC=Prediction_Error(log(SM.AICdf$raw_y),log(SM.AICdf$y)),
             S.AIC=Prediction_Error(log(S.AICdf$raw_y),log(S.AICdf$y)),
             PS.AIC=Prediction_Error(log(PS.AICdf$raw_y),log(PS.AICdf$y)),
             SM.lasso=Prediction_Error(log(SM.lassodf$raw_y),log(SM.lassodf$y)),
             S.lasso=Prediction_Error(log(S.lassodf$raw_y),log(S.lassodf$y)),
```

```
              PS.lasso=Prediction_Error(log(PS.lassodf$raw_y),log(PS.lassodf$y)),
              LASSO=Prediction_Error(log(lassodf$raw_y),log(lassodf$y)),
              ENET=Prediction_Error(log(enetdf$raw_y),log(enetdf$y)),
              ALASSO=Prediction_Error(log(alassodf$raw_y),log(alassodf$y)),
              SCAD=Prediction_Error(log(scaddf$raw_y),log(scaddf$y)),
              RIDGE=Prediction_Error(log(ridgedf$raw_y),log(ridgedf$y))
  )
  PE_star
  PE_values_bootstrap[i,] <- PE_star

  FMdf_100 <- FMdf[order(FMdf$raw_y, decreasing = TRUE), ][1:100, ]
  SM.BICdf_100 <- SM.BICdf[order(SM.BICdf$raw_y, decreasing = TRUE), ][1:100, ]
  S.BICdf_100 <- S.BICdf[order(S.BICdf$raw_y, decreasing = TRUE), ][1:100, ]
  PS.BICdf_100 <- PS.BICdf[order(PS.BICdf$raw_y, decreasing = TRUE), ][1:100, ]
  SM.AICdf_100 <- SM.AICdf[order(SM.AICdf$raw_y, decreasing = TRUE), ][1:100, ]
  S.AICdf_100 <- S.AICdf[order(S.AICdf$raw_y, decreasing = TRUE), ][1:100, ]
  PS.AICdf_100 <- PS.AICdf[order(PS.AICdf$raw_y, decreasing = TRUE), ][1:100, ]
  SM.lassodf_100 <- SM.lassodf[order(SM.lassodf$raw_y, decreasing = TRUE), ][1:100, ]
  S.lassodf_100 <- S.lassodf[order(S.lassodf$raw_y, decreasing = TRUE), ][1:100, ]
  PS.lassodf_100 <- PS.lassodf[order(PS.lassodf$raw_y, decreasing = TRUE), ][1:100, ]
  lassodf_100 <- lassodf[order(lassodf$raw_y, decreasing = TRUE), ][1:100, ]
  enetdf_100 <- enetdf[order(enetdf$raw_y, decreasing = TRUE), ][1:100, ]
  alassodf_100 <- alassodf[order(alassodf$raw_y, decreasing = TRUE), ][1:100, ]
  scaddf_100 <- scaddf[order(scaddf$raw_y, decreasing = TRUE), ][1:100, ]
  ridgedf_100 <- ridgedf[order(ridgedf$raw_y, decreasing = TRUE), ][1:100, ]
  #Calculate prediction errors of estimators for top 100 y values.
  PE_star_100 <- c(FM=Prediction_Error(log(FMdf_100$raw_y),log(FMdf_100$y)),
SM.BIC=Prediction_Error(log(SM.BICdf_100$raw_y),log(SM.BICdf_100$y)),
S.BIC=Prediction_Error(log(S.BICdf_100$raw_y),log(S.BICdf_100$y)),
 PS.BIC=Prediction_Error(log(PS.BICdf_100$raw_y),log(PS.BICdf_100$y)),
 SM.AIC=Prediction_Error(log(SM.AICdf_100$raw_y),log(SM.AICdf_100$y)),
 S.AIC=Prediction_Error(log(S.AICdf_100$raw_y),log(S.AICdf_100$y)),
 PS.AIC=Prediction_Error(log(PS.AICdf_100$raw_y),log(PS.AICdf_100$y)),
 SM.lasso=Prediction_Error(log(SM.lassodf_100$raw_y),log(SM.lassodf_100$y)),
 S.lasso=Prediction_Error(log(S.lassodf_100$raw_y),log(S.lassodf_100$y)),
 PS.lasso=Prediction_Error(log(PS.lassodf_100$raw_y),log(PS.lassodf_100$y)),
 LASSO=Prediction_Error(log(lassodf_100$raw_y),log(lassodf_100$y)),
 ENET=Prediction_Error(log(enetdf_100$raw_y),log(enetdf_100$y)),
 ALASSO=Prediction_Error(log(alassodf_100$raw_y),log(alassodf_100$y)),
 SCAD=Prediction_Error(log(scaddf_100$raw_y),log(scaddf_100$y)),
```

```
 RIDGE=Prediction_Error(log(ridgedf_100$raw_y),log(ridgedf_100$y))
  )
  PE_star_100
  PE100_values_bootstrap[i,] <- PE_star_100
}
PE_avg <- apply(PE_values_bootstrap,2,mean)
PE_std <- apply(PE_values_bootstrap,2,sd)
PE.result <- as.data.frame(cbind(PE_avg, PE_std))
# Add row names
row_names <- c("FW","SM.BIC","S.BIC","PS.BIC","SM.AIC","S.AIC","PS.AIC","SM.lasso",
"S.lasso","PS.lasso", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE.result) <- row_names
PE.result

PE100_avg <- apply(PE100_values_bootstrap,2,mean)
PE100_std <- apply(PE100_values_bootstrap,2,sd)
PE100.result <- as.data.frame(cbind(PE100_avg, PE100_std))
# Add row names
row_names <- c("FW","SM.BIC","S.BIC","PS.BIC","SM.AIC","S.AIC","PS.AIC","SM.lasso",
"S.lasso","PS.lasso", "LASSO","ENET","ALASSO","SCAD","RIDGE")
rownames(PE100.result) <- row_names
PE100.result
```

REFERENCES

[1] Mathew Abraham. "Studying the patterns and long-run dynamics in cryptocurrency prices". In: *Journal of Corporate Accounting & Finance* 31.3 (2020), pp. 98–113.

[2] S Ejaz Ahmed. *Penalty, shrinkage and pretest strategies: Variable selection and estimation*. Springer, 2014.

[3] Syed Ejaz Ahmed. "Shrinkage estimation of regression coefficients from censored data with multiple observations". In: *Empirical Bayes and Likelihood Inference*. Springer, 2001, pp. 103–120.

[4] Syed Ejaz Ahmed, Feryaal Ahmed, and Bahadir Yüzbaşı. *Post-shrinkage strategies in statistical and machine learning for high dimensional data*. Chapman and Hall/CRC, 2023.

[5] Erdinc Akyildirim, Ahmet Goncu, and Ahmet Sensoy. "Prediction of cryptocurrency returns using machine learning". In: *Annals of Operations Research* 297.1 (2021), pp. 3–36.

[6] Bashar Yaser Almansour, Muneer M Alshater, and Ammar Yaser Almansour. "Performance of ARCH and GARCH models in forecasting cryptocurrency market volatility". In: *Industrial Engineering & Management Systems* 20.2 (2021), pp. 130–139.

[7] Kolmogorov An. "Sulla determinazione empirica di una legge didistribuzione". In: *Giorn Dell'inst Ital Degli Att* 4 (1933), pp. 89–91.

[8] Mohammad Arashi, Yasin Asar, and Bahadır Yüzbaşı. "SLASSO: A scaled LASSO for multicollinear situations". In: *Journal of Statistical Computation and Simulation* 91.15 (2021), pp. 3170–3183.

[9] David Ardia et al. "Return and risk of pairs trading using a simulation-based bayesian procedure for predicting stable ratios of stock prices". In: *Econometrics* 4.1 (2016), p. 14.

[10] Mona Azadkia and Sourav Chatterjee. "A simple measure of conditional dependence". In: *The Annals of Statistics* 49.6 (2021), pp. 3070–3102.

[11] Manoj Bahuguna and Ravindra Khattree. "A generic all-purpose transformation for multivariate modeling through copulas". In: *International Journal of Data Science and Analytics* 10.1 (2020), pp. 1–23.

[12] Roy E Bailey. *The economics of financial markets*. Cambridge University Press, 2005.

[13] Benjamin M Blau. "Price dynamics and speculative trading in bitcoin". In: *Research in international Business and Finance* 41 (2017), pp. 493–499.

[14] Alexander Brauneis, Roland Mestel, and Erik Theissen. “What drives the liquidity of cryptocurrencies? A long-term analysis”. In: *Finance Research Letters* 39 (2021), p. 101537.

[15] Ngai Hang Chan and Ching Zong Wei. “Limiting distributions of least squares estimates of unstable autoregressive processes”. In: *The annals of Statistics* (1988), pp. 367–401.

[16] Sourav Chatterjee. “A new coefficient of correlation”. In: *Journal of the American Statistical Association* 116.536 (2021), pp. 2009–2022.

[17] Sourav Chatterjee and Susan Holmes. “XICOR: Association measurement through cross rank increments”. In: *R package version 0.3* 3 (2020).

[18] Eng-Tuck Cheah and John Fry. “Speculative bubbles in Bitcoin markets? An empirical investigation into the fundamental value of Bitcoin”. In: *Economics letters* 130 (2015), pp. 32–36.

[19] Jeffrey Chu et al. “GARCH modelling of cryptocurrencies”. In: *Journal of Risk and Financial Management* 10.4 (2017), p. 17.

[20] Christian Conrad, Anessa Custovic, and Eric Ghysels. “Long-and short-term cryptocurrency volatility components: A GARCH-MIDAS analysis”. In: *Journal of Risk and Financial Management* 11.2 (2018), p. 23.

[21] Fabian Dablander. *Causal Effect of Elon Musk Tweets on Dogecoin Price*. `https://fabiandablander.com/r/Causal-Doge.html`. 2024.

[22] David A Dickey and Wayne A Fuller. “Distribution of the estimators for autoregressive time series with a unit root”. In: *Journal of the American statistical association* 74.366a (1979), pp. 427–431.

[23] Abeer ElBahrawy et al. “Evolutionary dynamics of the cryptocurrency market”. In: *Royal Society open science* 4.11 (2017), p. 170623.

[24] Electricity Consumers Resource Council. *The Economic Impacts of the August 2003 Blackout*. Tech. rep. Washington, DC: Electricity Consumers Resource Council, Feb. 2004. URL: `https://www.nrc.gov/docs/ML1113/ML111300584.pdf`.

[25] Robert F Engle and Kenneth F Kroner. “Multivariate simultaneous generalized ARCH”. In: *Econometric theory* 11.1 (1995), pp. 122–150.

[26] Robert F Engle and Victor K Ng. “Measuring and testing the impact of news on volatility”. In: *The journal of finance* 48.5 (1993), pp. 1749–1778.

[27] Jianqing Fan and Runze Li. “Variable selection via nonconcave penalized likelihood and its oracle properties”. In: *Journal of the American statistical Association* 96.456 (2001), pp. 1348–1360.

[28] Tong Fang, Zhi Su, and Libo Yin. “Economic fundamentals or investor perceptions? The role of uncertainty in predicting long-term cryptocurrency volatility”. In: *International Review of Financial Analysis* 71 (2020), p. 101566.

[29] Miroslav Fil and Ladislav Kristoufek. “Pairs trading in cryptocurrency markets”. In: *Ieee Access* 8 (2020), pp. 172644–172651.

[30] Yahoo Finance. *Yahoo Finance*. `https://finance.yahoo.com/`.

[31] Thomas J Fisher and Colin M Gallagher. “New weighted portmanteau statistics for time series goodness of fit testing”. In: *Journal of the American Statistical Association* 107.498 (2012), pp. 777–787.

[32] Markus J Fülle et al. “BEKKs: An R Package for Estimation of Conditional Volatility of Multivariate Time Series”. In: *Journal of Statistical Software* 111 (2024), pp. 1–34.

[33] Xiaoli Gao, Syed Ejaz Ahmed, and Yang Feng. “Post selection shrinkage estimation for high-dimensional data analysis”. In: *Applied Stochastic Models in Business and Industry* 33.2 (2017), pp. 97–120.

[34] Alexios Ghalanos. “Introduction to the rugarch package.(Version 1.3-1)”. In: *Manuscript, http://cran. r-project. org/web/packages/rugarch. Accessed* 11.8 (2020).

[35] Sucharita Ghosh. “A new graphical tool to detect non-normality”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.4 (1996), pp. 691–702.

[36] Sascha Hägele. “Centralized exchanges vs. decentralized exchanges in cryptocurrency markets: A systematic literature review”. In: *Electronic Markets* 34.1 (2024), p. 33.

[37] Peter R Hansen and Asger Lunde. “A forecast comparison of volatility models: does anything beat a GARCH (1, 1)?” In: *Journal of applied econometrics* 20.7 (2005), pp. 873–889.

[38] Syed H Hasan et al. “A Novel Cryptocurrency Prediction Method Using Optimum CNN.” In: *Computers, Materials & Continua* 71.1 (2022).

[39] Investing.com. *Investing.com - Stock Market Quotes & Financial News*. `https://www.investing.com/`. 2024.

[40] William James and Charles Stein. “Estimation with quadratic loss”. In: *Breakthroughs in statistics: Foundations and basic theory*. Springer, 1992, pp. 443–460.

[41] Alboukadel Kassambara. *Practical guide to principal component methods in R: PCA, M (CA), FAMD, MFA, HCPC, factoextra*. Vol. 2. Sthda, 2017.

[42] Ravindra Khattree and Dayanand N Naik. “Multivariate data reduction and discrimination”. In: *SAS Institute, Cary, North Carolina* (2000).

[43] Ahmed M Khedr et al. “Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey”. In: *Intelligent Systems in Accounting, Finance and Management* 28.1 (2021), pp. 3–34.

[44] Taewook Kim and Ha Young Kim. “Optimizing the Pairs-Trading Strategy Using Deep Reinforcement Learning with Trading and Stop-Loss Boundaries”. In: *Complexity* 2019.1 (2019), p. 3582516.

[45] Sebastian Krantz et al. “dfms: Dynamic Factor Models”. In: *CRAN: Contributed Packages* 1 (2023), p. 357.

[46] Connor Lamon, Eric Nielsen, and Eric Redondo. “Cryptocurrency price prediction using news and social media sentiment”. In: *SMU Data Sci. Rev* 1.3 (2017), pp. 1–22.

[47] Jim Liew et al. “Cryptocurrency investing examined”. In: *The Journal of The British Blockchain Association* 2.2 (2019), pp. 1–12. DOI: 10.31585/jbba-2-2-(2)2019.

[48] Jiatao Liu et al. “Factor structure in cryptocurrency returns and volatility”. In: *Available at SSRN 3389152* (2019).

[49] Jinan Liu and Apostolos Serletis. “Volatility in the cryptocurrency market”. In: *Open Economies Review* 30.4 (2019), pp. 779–811.

[50] Kanti V Mardia. “Measures of multivariate skewness and kurtosis with applications”. In: *Biometrika* 57.3 (1970), pp. 519–530.

[51] Kanti V Mardia. “Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies”. In: *Sankhyā: The Indian Journal of Statistics, Series B* (1974), pp. 115–128.

[52] Monia Milutinović. “Cryptocurrency”. In: *Economics – Journal for Economic Theory and Practice and Social Issues* 1 (2018), pp. 105–122.

[53] Anthony Ngunyi, Simon Mundia, and Cyprian Omari. “Modelling volatility dynamics of cryptocurrencies using GARCH models”. In: *Journal of Mathematical Finance* 9 (2019), pp. 591–615.

[54] Chun-Xiao Nie. “Correlation dynamics in the cryptocurrency market based on dimensionality reduction analysis”. In: *Physica A: Statistical Mechanics and its Applications* 554 (2020), p. 124702.

[55] Jukka Nyblom. “Testing for the constancy of parameters over time”. In: *Journal of the American Statistical Association* 84.405 (1989), pp. 223–230.

[56] Franz C Palm. “GARCH models of volatility”. In: *Handbook of statistics* 14 (1996), pp. 209–240.

[57] Marcelo Scherer Perlin et al. “A garch tutorial with r”. In: *Revista de Administração Contemporânea* 25 (2020), e200088.

[58] Peter CB Phillips. “Time series regression with a unit root”. In: *Econometrica: Journal of the Econometric Society* (1987), pp. 277–301.

[59] Andrew Pole. *Statistical arbitrage: algorithmic trading insights and techniques*. John Wiley & Sons, 2011.

[60] TG Saji. “Pairs trading in cryptocurrency market: A long-short story”. In: *Available at SSRN 5135627* (2021).

[61] Soham Sarkar and Anil K Ghosh. “Some multivariate tests of independence based on ranks of nearest neighbors”. In: *Technometrics* 60.1 (2018), pp. 101–111.

[62] Agam Shah, Yagnesh Chauhan, and Bhaskar Chaudhury. “Principal component analysis based construction and evaluation of cryptocurrency index”. In: *Expert systems with applications* 163 (2021), p. 113796.

[63] Samuel S Shapiro, Martin B Wilk, and Hwei J Chen. “A comparative study of various tests for normality”. In: *Journal of the American statistical association* 63.324 (1968), pp. 1343–1372.

[64] Samuel Sanford Shapiro and Martin B Wilk. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52.3-4 (1965), pp. 591–611.

[65] James H Stock and Mark W Watson. “Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics”. In: *Handbook of macroeconomics*. Vol. 2. Elsevier, 2016, pp. 415–525.

[66] Masood Tadi and Irina Kortchemski. “Evaluation of dynamic cointegration-based pairs trading strategy in the cryptocurrency market”. In: *Studies in Economics and Finance* 38.5 (2021), pp. 1054–1075.

[67] Huong NQ Tran and Ravindra Khattree. “A Revisit to the Graphical Examination of Normality, Multinormality and Other Probability Distributions: H. Tran and R. Khattree”. In: *Sankhya B* (2025), pp. 1–66.

[68] Huong NQ Tran and Ravindra Khattree. “Visualization of Multivariate Normality: A Cumulant Generating Function Approach”. In: *Sankhya A* (2025), pp. 1–46.

[69] Ruey S Tsay. *Analysis of financial time series*. John wiley & sons, 2010.

[70] Ganapathy Vidyamurthy. *Pairs Trading: quantitative methods and analysis*. John Wiley & Sons, 2004.

[71] Marcin Wątorek et al. “Multiscale characteristics of the emerging global cryptocurrency market”. In: *Physics Reports* 901 (2021), pp. 1–82.

[72] Hui Zou. “The adaptive lasso and its oracle properties”. In: *Journal of the American statistical association* 101.476 (2006), pp. 1418–1429.

[73] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2 (2005), pp. 301–320.