CAPM Assignment

Tom O Nuallain
20365261

FIN30200
University College Dublin

October 15, 2023

1 Introduction

A fundamental idea of modern finance is that an investor needs a financial incentive to take a risk. In other words, the expected return on a risky investment, R, must exceed the return on a safe, or risk-free, investment, R_f. Thus, the expected excess return, $R - R_f$, on a risky investment, like owning stock in a company, should be positive. At first, it might seem like the risk of an asset should be measured by its variance. Much of that risk, however, can be reduced by holding other assets in a "portfolio" (i.e. by diversifying your financial holdings). This means that the right way to measure the risk of an asset is not by its variance but rather by its covariance with the market. The Capital Asset Pricing Model (CAPM) formalises this idea. According to this model, the expected excess return on an asset is proportional to the expected excess return on a portfolio of all available assets ("the market portfolio"). The CAPM says:

$$(R_{it} - R_{ft}) = \alpha_j + \beta_j(R_{mt} - R_{ft})$$

where R_{it} is the expected return of the asset j, R_{ft} measures the expected of return of a risk-free asset during period t (the Federal Funds Rate, FFR), and R_{mt} is the expected return on the market portfolio during period t (SP500).

According to the CAPM, an asset with a $\beta < 1$ has less risk than the market portfolio and therefore has a lower expected excess return than the market portfolio. Meanwhile, an asset with $\beta > 1$ is riskier than the market portfolio and thus commands a higher expected excess return.

The purpose of this assignment is to use the CAPM to model two cryptocurrencies, Bitcoin (BTC) and Ethereum (ETH). For both of these assets, daily close prices were sourced from Yahoo finance over a period from the 18th September 2014 to the 31st August for Bitcoin, and 9th November 2017 to the 31st August 2023 for Ethereum. The risk free interest rate and S&P 500 daily data is sourced from the Federal Reserve of St. Louis for the same periods.

2 Simple Linear Regression to Model CAPM

To begin, we will run a simple linear regression model for both BTC and ETH.

The classical linear regression model is a statistical technique used to model the relationship between a dependent variable (Y) and one or more independent variables ($X_1, X_2, \ldots, X_p$). It assumes a linear relationship between the independent variables and the dependent variable. In our case, we want to model the relationship between the log excess returns of a cryptocurrency and the log excess returns of the market. The general model can be expressed as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon \tag{1}$$

Where:

$$\begin{aligned}
Y &: \text{Dependent Variable} \\
X_i &: \text{Independent Variables (Predictors)} \\
\beta_0 &: \text{Intercept (Y-intercept)} \\
\beta_i &: \text{Coefficients of Independent Variables} \\
\varepsilon &: \text{Error Term (Residuals)}
\end{aligned}$$

In our case, we will just use a simple linear regression, where we are only interested in an intercept (α) and a slope (β) coefficient, i.e.

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{2}$$

Where:

$$\begin{aligned}
Y &: \text{Dependent Variable } (R_{it} - R_{ft}) \\
X &: \text{Independent Variable } (R_{mt} - R_{ft}) \\
\beta_0 &: \text{Intercept } (\alpha) \\
\beta_1 &: \text{Slope } (\beta) \\
\varepsilon &: \text{Error Term (Residual)}
\end{aligned}$$

We can derive the estimates for β_0 and β_1 as follows:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \\
\hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}
\end{aligned}$$

where $\bar{X}$ is the mean of X and $\bar{Y}$ is the mean of Y.

2.0.1 Model Assuptions

The classical linear regression model relies on several key assumptions (Brooks, 2008):

Linearity

The relationship between the dependent variable Y and the independent variables X_i is assumed to be linear.

Zero Mean of Errors

The errors (ε) have a mean of zero. In mathematical notation:

$$\mathbb{E}(\varepsilon) = 0 \tag{3}$$

Constant Variance of Errors (Homoskedasticity)

The variance of the errors (ε) is constant and finite over all values of the independent variables X_i. This assumption is often referred to as homoskedasticity and can be expressed as:

$$\text{Var}(\varepsilon) = \sigma^2 \tag{4}$$

Independence of Errors

The errors (ε_i) are linearly independent of one another. In other words, the error in one observation is not correlated with the error in any other observation. This can be expressed as:

$$\text{Cov}(\varepsilon_i, \varepsilon_j) = 0 \text{ for } i \neq j \tag{5}$$

No Relationship Between Errors and Predictors

There is no relationship between the errors (ε) and the corresponding independent variables X_i. This means that the errors are not systematically related to the predictors. Mathematically:

$$\text{Cov}(\varepsilon_i, X_i) = 0 \text{ for all } i \tag{6}$$

These assumptions are crucial for the classical linear regression model to provide unbiased and efficient estimates of the model parameters (β_i) and valid hypothesis tests.

2.1 Transformation of Data

As cryptocurrencies trade 24/7, including on weekend days, but the S&P 500 only trades on weekdays, there were a number of observations where data existed for Bitcoin and Ethereum but not for the S&P 500. I opted to remove these weekend dates, so that we only have observations in the data set where both the cryptocurrency and the S&P 500

traded. While this results in a lower number of observations, and potentially loses some information about cryptocurrency returns, the alternative would have been to either have the S&P 500 return as zero for these days, or try to interpolate a return based on the Friday close and Monday open. I figured this was less accurate and could have an adversarial affect on the CAPM estimation by using data points which did not actually occur.

2.1.1 Log Transformation

Before we run our model, we must perform some transformations on our data set. For both crypto currencies and for the S&P 500, we are interested in obtaining the log daily returns, i.e.

$$R_{i,t} = \log\left(\frac{P_{i,t}}{P_{i,t-1}}\right) \times 100 \tag{7}$$

There are a few reasons why we want to log transform our data, and I will highlight some of them below (AKdemy (2021)):

The log difference is approximating percent change
It can be shown that using log returns approximates the continuously compounded percent change in asset prices over time. This approximation aligns well with the concept of expected returns and risk in the CAPM, which is expressed as a percentage.

Data is more likely normally distributed
In the CAPM, it's often assumed that returns follow a normal distribution or close to normal. This assumption simplifies the estimation of parameters and is useful when analyzing the relationship between expected returns and systematic risk. Using log returns helps make the data distribution closer to normal, making CAPM modeling assumptions more plausible, especially for a linear regression.

Data is more likely homoskedastic
Homoskedasticity implies that the variance of returns is constant over time. In the context of CAPM, where you're interested in risk assessment and the relationship between returns and systematic risk (beta), maintaining constant variance is important. Log returns tend to exhibit this property more effectively, which is desirable when estimating the CAPM.

2.1.2 Calculation of Dependent and Independent Variables

Now, for both BTC and ETH we can calculate the dependent variable for our model, $(R_{it} - R_{ft})$, for each observation by subtracting the risk free return from the return on the cryptocurrency. Next, we can calculate the independent variable, $(R_{mt} - R_{ft})$, by subtracting the risk free return from the return on the S&P500 at each observation. Now, we are ready to regress our dependent variable on our independent variable, for both BTC and ETH.

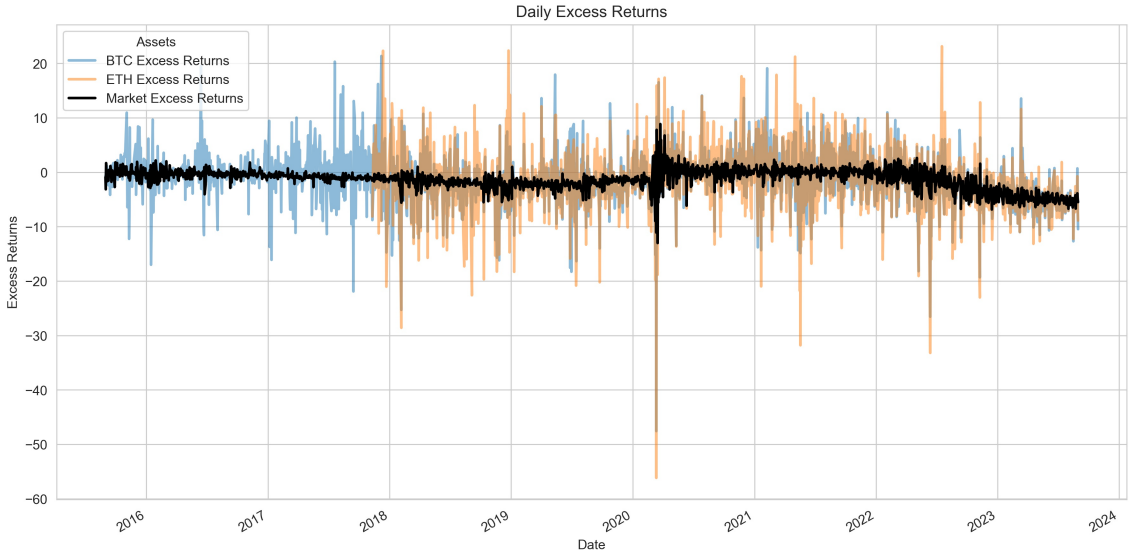


Figure 1: Daily Excess Returns (BTC, ETH, S&P500)

2.2 Model Results

Table 1: Linear Regression Results (BTC and ETH)

	BTC		ETH	
	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\alpha}$	$\hat{\beta}$
coefficient	0.1682	1.0094***	0.3674**	1.2264***
$\hat{SE}$	0.119	0.053	0.184	0.071
R^2	0.159		0.174	
Adj. R^2	0.159		0.174	
F-stat	367.2		296	
Prob(F-stat)	4.34e-75		2.48e-60	
Log-likelihood	-5578.2		-4373.9	
AIC	1.116e+04		8752	
BIC	1.117e+04		8762	
Omnibus	277.877		245.075	
Prob(Omnibus)	0.000		0.000	
Durbin-Watson	1.987		2.052	
Jarque-Bera (JB)	3384.188		2226.082	
Prob(JB)	0.00		0.00	
Skew	-0.195		-0.526	
Kurtosis	9.457		9.076	

Note: * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

We can now examine the results of our simple linear regression models for BTC and ETH (Table 1). Also, see Figure 2 for model plots and residuals.

2.2.1 Coefficient Estimates - Interpretation and Significance

For each coefficient estimate in both models, $\hat{\alpha}$ and $\hat{\beta}$, or in the general case $\hat{\beta}_0$, and $\hat{\beta}_1$, we want to test the significance of the estimate using three different methods.

Method 1: T-Test

Hypotheses

The null hypothesis (H_0) and the alternative hypothesis (H_1) for the t-test are as follows:

$$\begin{cases} H_0 : \beta_i = 0 \\ H_1 : \beta_i \neq 0 \end{cases}$$

Test Statistic

Calculate the t-statistic ($t_{\text{empirical}}$) using the formula:

$$t_{\text{empirical}} = \frac{\hat{\beta}_i - 0}{SE(\hat{\beta}_i)}$$

Critical Value

Determine the critical value (t_{critical}) from the t-distribution table for a given significance level (α) and degrees of freedom (df).

Conclusion

If $|t_{\text{empirical}}| > t_{\text{critical}}$, reject the null hypothesis (H_0) and conclude that the coefficient estimate ($\hat{\beta}_i$) is statistically significant.

Method 2: P-Value Test

Hypotheses

The null hypothesis (H_0) and the alternative hypothesis (H_1) for the p-value test are the same as in Method 1.

P-Value Calculation

Calculate the p-value (P) associated with the coefficient estimate ($\hat{\beta}_i$) from the t-distribution.

Conclusion

If $P < \alpha$, reject the null hypothesis (H_0) and conclude that the coefficient estimate ($\hat{\beta}_i$) is statistically significant.

Method 3: Confidence Interval Test

Confidence Interval Calculation

Calculate a confidence interval for the coefficient estimate ($\hat{\beta}_i$). This interval will look

like:

$$CI(\beta_i) = \hat{\beta}_i \pm t_{\alpha/2,(n-p)} \cdot SE(\hat{\beta}_i) \tag{8}$$

Where:

- $\hat{\beta}_i$ is the estimated coefficient value.
- $t_{\frac{\alpha}{2},(n-p)}$ is the critical value from the t-distribution with $(n-p)$ degrees of freedom and $\frac{\alpha}{2}$ significance level.
- α represents the desired confidence level (e.g., 0.05 for a 95% confidence interval).
- $SE(\hat{\beta}_i)$ is the standard error of the coefficient estimate.

Conclusion

If the confidence interval does not include zero, conclude that the coefficient estimate ($\hat{\beta}_i$) is statistically significant.

Table 2: Hypothesis Test Results for Coefficient Significance (BTC, $\alpha = 0.05$)

Coefficient	**Empirical T**	**Critical T**	**P-Value Test**	**Conf. Interval**	**Result**
$\hat{\alpha}$	1.412	1.96	0.158	[-0.065, 0.402]	fail to reject H_0
$\hat{\beta}$	19.162	1.96	0.000	[0.906, 1.113]	reject H_0

BTC Model Coefficients

Interpretation

For the BTC Model, we obtain an $\hat{\alpha}$ coefficient of 0.1682. The economic interpretation of this is that this figure represents the expected log excess return of Bitcoin above the risk free rate when the log excess return of the market above the risk free rate is zero. This indicates that we expect the asset to outperform the market when the market return is zero. In other words, it suggests the asset has provided a positive return beyond what would be expected from its beta. We also have a $\hat{\beta}$ coefficient of 1.0094. The beta measures the asset's sensitivity to market movements. An economic interpretation of this is that for a one percentage point rise (fall) in the excess return of the market, we expect the excess return of Bitcoin to rise (fall) by 1.0094 percentage points. In this case, the β being very close to one indicates the asset moves in line with the market. The beta measures the asset's sensitivity to market movements.

Significance

We will refer to Table 2. Firstly, using method 1 we will compare our empirical t-stat of 1.412 for the $\hat{\alpha}$ coefficient estimate against a t-statistic of 1.96 for a level of significance

equal to 0.05. This empirical t-stat corresponds to a p-value of 0.158. Also, 0 is included in the 95% confidence interval. Because of this, and since $|t_{\text{empirical}}| \not> t_{\text{critical}}$, and $P > 0.05$, we fail to reject to reject H_0, and as such we do not find this coefficient estimate to be statistically significant. The CAPM assumption that $\alpha = 0$ holds for the BTC model.
Next, for the $\hat{\beta}$ coefficient we get an empirical t-stat of 19.162, well above the critical value of 1.96. This t-stat corresponds to a very low p-value, close to zero. We can also see that 0 is not included in the 95% confidence interval for our estimate. Therefore, since $|t_{\text{empirical}}| > t_{\text{critical}}$, and $P < 0.05$, we reject the null hypothesis and find that $\hat{\beta}$ is significantly different from zero.

Table 3: Hypothesis Test Results for Coefficient Significance (ETH, $\alpha = 0.05$)

Coefficient	**Empirical T**	**Critical T**	**P-Value Test**	**Conf. Interval**	**Result**
$\hat{\alpha}$	1.991	1.96	0.047	[0.006, 0.729]	reject H_0
$\hat{\beta}$	17.203	1.96	0.000	[1.087, 1.366]	reject H_0

ETH Model Coefficients

Interpretation

For the ETH Model, we have an $\hat{\alpha}$ coefficient of 0.3674. We can interpret this as follows: this figure represents the expected log excess return of Bitcoin above the risk free rate when the log excess return of the market above the risk free rate is zero. Given the positive alpha, we expect Ethereum to outperform the market when the market excess return is zero. In other words, it suggests the asset has provided a positive return beyond what would be expected from its beta. We also have a $\hat{\beta}$ coefficient of 1.2264. The beta measures the asset's sensitivity to market movements. An economic interpretation of this is that for a one percentage point rise (fall) in the excess return of the market, we expect the excess return of Bitcoin to rise (fall) by 1.2264 percentage points. In this case, the β being slightly above one indicates a move in the market will cause a disproportionately greater move in this asset. The beta measures the asset's sensitivity to market movements.

Significance

We will refer to Table 3. Using method 1 we will compare our empirical t-stat of 1.991 for the $\hat{\alpha}$ coefficient estimate against a t-statistic of 1.96 for a level of significance equal to 0.05. This empirical t-stat corresponds to a p-value of 0.047, just under 0.05. Also, 0 is not included in the 95% confidence interval. Thus, since $|t_{\text{empirical}}| > t_{\text{critical}}$, and $P < 0.05$, we can reject H_0 in favour of H_1, and as such we find this coefficient estimate to be statistically significant at the 5% level of significance. Since we have found that $\alpha \neq 0$, we reject the CAPM null hypothesis.
Next, for the $\hat{\beta}$ coefficient we get an empirical t-stat of 17.203, above the critical value of 1.96. This t-stat corresponds to a very low p-value, close to zero. We can also see that 0 is not included in the 95% confidence interval for our estimate. Therefore, since $|t_{\text{empirical}}| >$

t_{critical}, and $P < 0.05$, we reject the null hypothesis and find that $\hat{\beta}$ is significantly different from zero.

2.2.2 Other Statistical Indicators and Results

R-squared

In simple linear regression, the R-squared value, denoted as R^2, is a measure of how well the linear regression model fits the data. It represents the proportion of the variance in the dependent variable that is explained by the independent variable.

The formula to calculate R^2 is as follows:

$$R^2 = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Where:

- Y_i is the observed value of the dependent variable,
- $\hat{Y}_i$ is the predicted value of the dependent variable from the regression model,
- $\bar{Y}$ is the mean of the observed values, and
- n is the number of data points.

Interpretation:

- R^2 ranges from 0 to 1.
- An R^2 value close to 1 indicates that the regression model explains a large portion of the variance in the dependent variable, suggesting a good fit.
- An R^2 value close to 0 means that the regression model does not explain much of the variance, indicating a poor fit.

For the BTC model, we saw in Table 1 that the model had an R^2 value of 0.159. This is quite close low, and indicates that the excess return on the market explains just 15.9% of the variation of the excess return of Bitcoin, which is a poor fit.

For the ETH model, we saw in Table 1 that the model had an R^2 value of 0.174. This is again quite low, and although slightly better, still indicates that just 17.4% of the variation of excess return on Ethereum can be explained by the excess return of the market.

F-statistic

The F-statistic is a statistical measure used in the context of linear regression to assess the overall goodness-of-fit of a regression model. It helps determine whether the model, as a whole, is statistically significant in explaining the variance in the dependent variable.

The F-statistic is derived from the analysis of variance (ANOVA) and is a ratio of two variances. The F-statistic measures the ratio of the explained variance (variance due to the regression model) to the unexplained variance (residual variance). In other words, it assesses whether the regression model provides a better fit to the data than a model with no predictors. The hypotheses associated with the F-statistic are as follows:

- Null Hypothesis (H_0): $\beta_1 = \beta_2 = 0$
  The restricted model (typically a model with no predictors) is as good as or better than the unrestricted model (the full regression model with predictors).
- Alternative Hypothesis (H_1): $\beta_j \neq 0$, for at least one value of j
  The unrestricted model is significantly better than the restricted model.

The restricted model (M_0) is a simplified version of the regression model that includes only an intercept term (no predictors). The unrestricted model (M_1) is the full regression model with all the predictors of interest. The F-statistic is used to compare these two models. To test the hypotheses, we calculate the F-statistic (F-empirical) from the data. It is calculated as follows:

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

Where:

- TSS is the total sum of squares, which is the sum of the squared differences between each observed dependent variable value(Y_i) and the mean of the dependent variable ($\bar{Y}$).
- RSS is the residual sum of squares, which is the sum of the squared differences between the observed dependent variable values (Y_i) and the predicted values from the regression model ($\hat{Y}_i$).
- p is the number of predictors (in our case, it's 1, representing the slope).
- n is the number of data points.

We also determine the F-critical value from the F-distribution with degrees of freedom associated with the models. If the F-empirical value is greater than the F-critical value, we reject the null hypothesis, indicating that the unrestricted model is a better fit. Additionally, we can calculate the p-value associated with the F-empirical value. A small p-value suggests strong evidence against the null hypothesis and supports the conclusion that the unrestricted model is superior.
For the BTC model, an F-statistic of 367.2 is obtained. We can use the F-table to find the critical value for significance level $\alpha = 0.05$ and $\mathscr{F}(2-1, 1941-2)$. We then obtain a critical f-value of 3.85. Our f-stat is larger (367.2) and corresponds to a p-value of 4.34e-75, which is very small. As such, we can reject the null hypothesis and conclude that at

least one of our coefficient estimates is significant.
For the ETH model, our critical f-value for significance level $\alpha = 0.05$ and $\mathscr{F}(2-1, 1405-2)$ will also be 3.85, which is well below the f-statistic of 296 that we get from the model. This corresponds to a p-value of 2.48e-60, and since this is well below 0.05, we can reject the null hypothesis in favour of the alternative, and conclude that at least one of our coefficient estimates is significant. We can also say that the model we estimated is significantly better than a model which just tries to use the intercept.

Log Likelihood

$$\mathscr{L}(\beta_0, \beta_1, \sigma^2, y, x) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2 \tag{9}$$

The log-likelihood output is a measure that quantifies how well a statistical model fits the observed data. Specifically, it helps assess the goodness of fit of a model and in simple terms, the it represents the likelihood of observing the actual data points, given the parameters estimated by the regression model. This quantifies how well a model aligns with the actual data, with higher values indicating a better fit. We will use log-liklihoods to compare models in Section 5.0.2.

Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC)
AIC and BIC are other statistical measures used in linear regression to evaluate the goodness of fit and assist in model selection. They help with comparing and choosing between different models, especially when deciding how many parameters to include, as these criteria will penalise models with too many parameters to prevent over fitting.

AIC considers the trade-off between model fit and model complexity by incorporating the likelihood, number of parameters, and sample size. On the other hand, BIC introduces a stronger penalty for model complexity, prioritizing simpler models

In our case, we can see that the ETH model has an AIC and BIC of 8752 and 8762, while the BTC model has values of 1.116e+04 and 1.117e+04. Later, in Section 5.0.2, we will compare these figures to a regression model which uses no intercept coefficient.

Omnibus
The omnibus test is similar to the Jarque-Bera test, and was created by D'Agostino and Belanger (D'Agostino, 1971, D'Agostino, Belanger, & D'Agostino, 1990). A very low p-value here indicates that the residuals in both models are likely not normally distributed, but we will discuss the normality of residuals more below in Section 2.3.1

Durbin-Watson See Section 2.3.2

Jarque-Bera See Section 2.3.1

Skewness
The skewness of a random variable X is defined as:

$$\text{Skewness} = \frac{\mathbb{E}[(X-\mu)^3]}{\sigma^3}$$

Skewness measures the asymmetry of the probability distribution of X. For the BTC model, we have a skew figure of -0.195, while the ETH model has a skew figure of -0.526. When skewness is negative, the tail on the left side is longer or fatter, indicating left-skewness. For normally distributed residuals, we would expect skewness of 0. Skewness in residuals indicates a potential problem with the assumption of normally distributed errors, and suggests that a model may not capture all the underlying patterns in the data, leading to skewed residuals.

Kurtosis
The kurtosis of a random variable X is defined as:

$$\text{Kurtosis} = \frac{\mathbb{E}[(X-\mu)^4]}{\sigma^4}$$

Kurtosis measures the "tailedness" of the probability distribution of X. For the BTC model, we have kurtosis figure of 9.457, while the ETH model has kurtosis of 9.076. When kurtosis is greater than, it is called "leptokurtic", and indicates heavier tails than a normal distribution. Kurtosis for a normal distribution would be 3. Excess kurtosis in residuals can reveal information about the tails of the distribution, and, specifically, leptokurtic residuals may indicate the presence of outliers or extreme values that are not accounted for by the model.

2.3 Analysis of Residuals

In order to assess the validity of our models, we must analyse our residuals. In particular, we are interested in examining the following:

- Check if residuals are normally distributed
- Check if residuals are serially correlated
- Check if residuals are heteroskedastic

2.3.1 Normality of Residuals

To begin, we can visually inspect the plotted histogram of residuals, along with the overlaid normal distribution in Figure 2. Here, the normal distribution represents the distribution $N \sim (\mu, \sigma)$, where μ, σ represent the mean and variance of the residuals. Upon

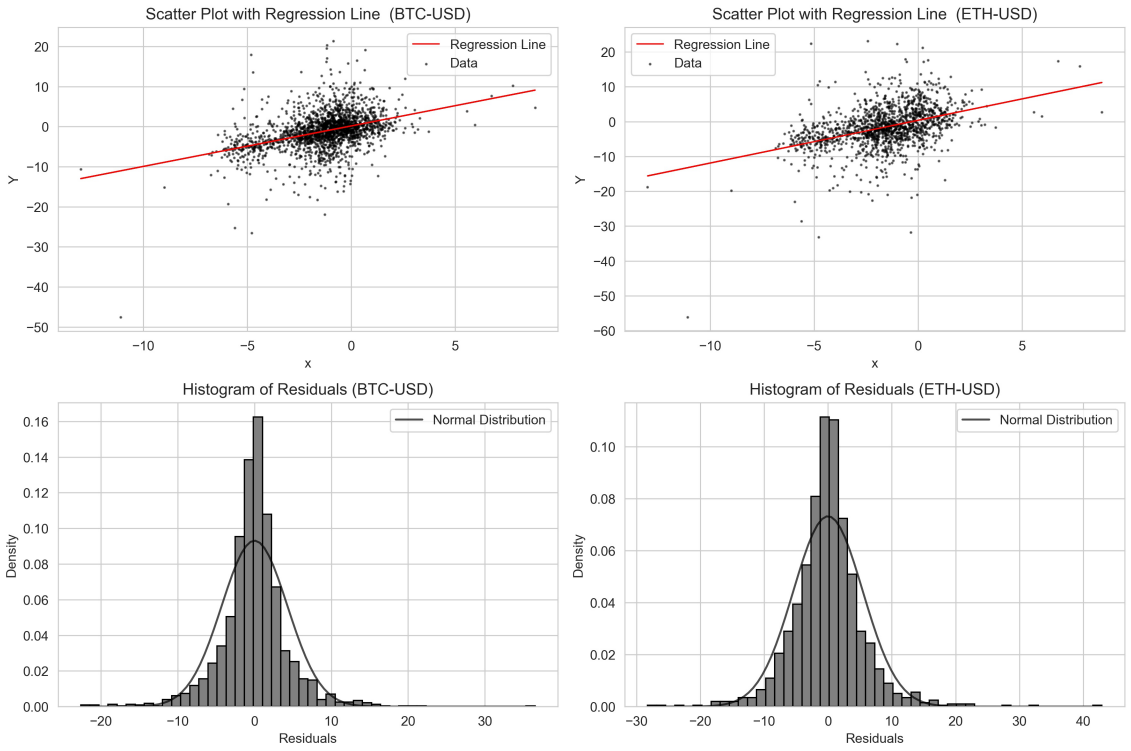


Figure 2: Simple Linear Regression and Residual Plots for BTC and ETH.

inspection of the BTC model, it does not appear that our residuals are normally distributed, as they do not appear particularly bell-shaped. The result is similar when looking at the ETH model. We can also use tests such as the Jarque-Bera test and the Anderson-Darling test, to check for normality of a distribution. Using multiple tests can provide a more robust evaluation of the data's normality.

Jarque-Bera Test for Normality

The Jarque-Bera test is used to assess whether a given dataset follows a normal distribution. It evaluates the normality of a dataset based on its sample skewness and sample kurtosis. We will use this test for both models, to test the hypothesis that are residuals are normally distributed.

Hypotheses

The hypotheses associated with the Jarque-Bera test are as follows:

- Null Hypothesis (H_0): The data is normally distributed (follows a Gaussian distribution).
- Alternative Hypothesis (H_1): The data is not normally distributed; it deviates from a Gaussian distribution.

Calculation

The Jarque-Bera statistic (JB) is calculated as follows:

$$JB = \frac{n}{6}\left(S^2 + \frac{1}{4}(K-3)^2\right)$$

Where:

n is the sample size.

S is the sample skewness.

K is the sample kurtosis.

To test the null hypothesis (H_0), the calculated JB statistic is compared to a critical value from the chi-squared (χ^2) distribution with 2 degrees of freedom, typically at a chosen significance level (e.g., 0.05). If JB is greater than the critical value, the null hypothesis is rejected, indicating that the data significantly deviates from a normal distribution.
For the BTC model, we obtain a Jarque-Bera statistic of 3384.188, which corresponds to a p-value of 0.00. Therefore, we find that the residuals deviate significantly from a normal distribution.
For the ETH model, we obtain a Jarque-Bera statistic of 2226.082, which corresponds to a p-value of 0.00. Again, we find that the residuals deviate significantly from a normal distribution, although perhaps to a lesser degree than the BTC model.

Anderson-Darling Test for Normality

The Anderson-Darling test is used to assess whether a given data set follows a particular probability distribution, such as the normal distribution(Anderson and Darling (1952)). It is a goodness-of-fit test that measures how well the observed data fits the theoretical distribution. In the case of the Anderson-Darling test for normality, it checks whether the data follows a normal distribution.

Calculation The Anderson-Darling statistic (A^2) is computed as follows:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}[2i-1]\left(\ln\left(F(x_i)\right) + \ln\left(1 - F(x_{n+1-i})\right)\right)$$

Where:

n is the sample size,

x_i represents the ordered values of the data,

$F(x_i)$ is the empirical cumulative distribution function (ECDF) of the data at point x_i,

ln denotes the natural logarithm.

The null and alternative hypotheses assosciated with the Anderson-Darling test are the same as desribed above for the Jarque-Bera test.

To perform hypothesis testing, we compute the Anderson-Darling statistic (A^2) from the data. We then compare this statistic to critical values from the Anderson-Darling distribution or use a significance level (α) to determine whether to reject the null hypothesis. If the calculated statistic is greater than the critical value or if the p-value is less than α, we reject the null hypothesis and conclude that the data do not follow a normal distribution. In the case of the BTC model, we obtain an Anderson-Darling statistic of 44.47, and the critical value for $\alpha = 0.05$ is 0.79. Since our statistic is larger than our critical value, we can reject the null hypothesis in favour of the alternative hypothesis, which is to say that the data do not follow a normal distribution.
For the ETH model, we obtain an Anderson-Darling statistic of 17.93, which again is greater than the critical value of 0.785, so we can reject the null hypothesis, and again conclude that the data do not follow a normal distribution.
To conclude, we find that residuals in both models are not normally distributed, which violates one of the assumptions of the classical linear regression model outlined in Section 2.0.1

2.3.2 Serial Correlation of Residuals

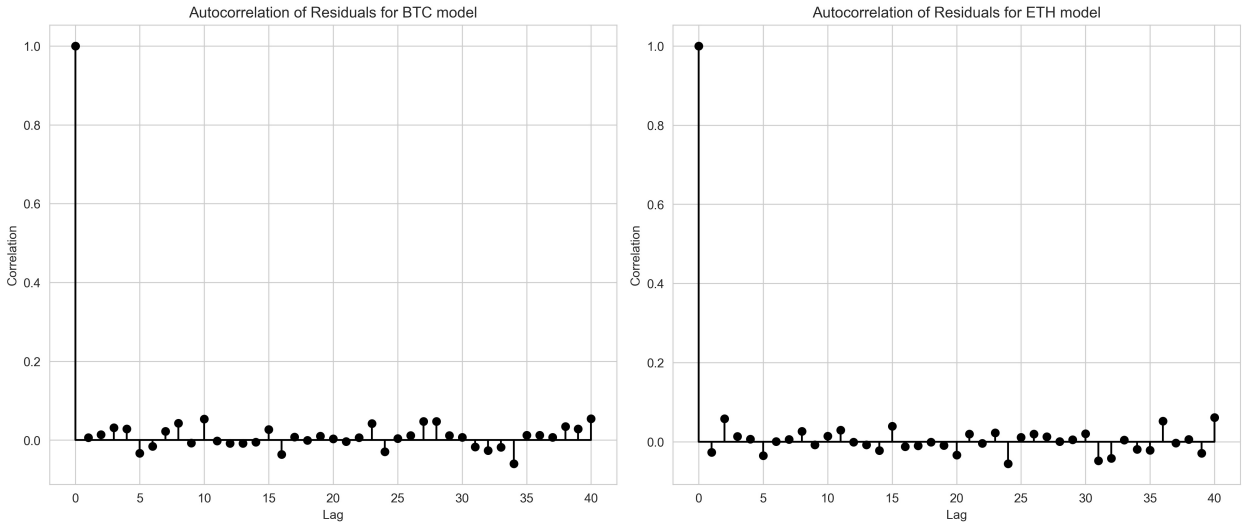


Figure 3: Autocorrelation of Residuals for BTC and ETH.

Next, we will examine whether or not the residuals are serially correlated. The first thing we can look at is autocorrelation of residuals for both models. The ACF at lag k for residuals ε is calculated as:

$$\rho_k = \frac{\text{Cov}(\hat{\varepsilon}_t, \hat{\varepsilon}_{t-k})}{\text{Var}(\hat{\varepsilon}_t)}$$

Where:

- k is the lag at which the ACF is calculated.
- $\text{Cov}(\hat{\varepsilon}_t, \hat{\varepsilon}_{t-k})$: Covariance between $\hat{\varepsilon}_t$ and $\hat{\varepsilon}_{t-k}$ for lag k
- $\text{Var}(\hat{\varepsilon}_t)$: Variance of $\hat{\varepsilon}_t$
- $\text{Var}(\hat{\varepsilon}_{t-k})$: Variance of $\hat{\varepsilon}_{t-k}$

As we can see from Figure 3, the ACF values are near zero for both the BTC and ETH models, which indicates little or no linear association. This implies that data points at those lags are relatively independent.

Breusch-Godfrey Test

One further test we can use to look for serial correlation in the residuals is the Breusch-Godfrey test (Breusch, 1978 & Godfrey, 1978). It is a general test for autocorrelation in the residuals, and is used when there is suspicion of autocorrelation beyond the first lag. Autocorrelation occurs when the error terms in a regression model are correlated with each other over time. This violates one of the key assumptions of ordinary least squares (OLS) regression, which assumes that the error terms are independently and identically distributed.

Performing the Test

To conduct the Breusch-Godfrey test, we followed these steps (Brooks, 2008):

1. Fit an auxiliary regression model, also known as the "auxiliary regression," where you regress the residuals ($\hat{\varepsilon}_t$) on lagged values of the independent variables (lags of $X_{1t}, X_{2t}, \ldots, X_{kt}$) and any other relevant covariates. The auxiliary regression for this given by:

$$\hat{\varepsilon}_t = \alpha_0 + \alpha_1 \hat{\varepsilon}_{t-1} + \alpha_2 \hat{\varepsilon}_{t-2} + \ldots + \alpha_p \hat{\varepsilon}_{t-p} + \gamma_1 X_{1t} + \gamma_2 X_{2t} + \ldots + \gamma_k X_{kt} + \text{other covariates} + u_t \tag{10}$$

   Where:

$$\begin{aligned} \hat{\varepsilon}_t &\text{ is the estimated residual at time } t \\ \alpha_0, \alpha_1, \ldots, \alpha_p &\text{ are parameters to be estimated} \\ \gamma_1, \gamma_2, \ldots, \gamma_k &\text{ are coefficients for lagged independent variables} \\ u_t &\text{ is the error term of the auxiliary regression} \end{aligned}$$

2. Compute the LM (Lagrange Multiplier) statistic for the auxiliary regression. The LM statistic tests whether the lagged residuals ($\hat{\varepsilon}_{t-1}, \hat{\varepsilon}_{t-2}, \ldots, \hat{\varepsilon}_{t-p}$) and the independent variables are jointly significant in explaining the current residuals ($\hat{\varepsilon}_t$).

3. The LM statistic follows a chi-squared distribution with degrees of freedom equal to the number of lagged residuals and independent variables in the auxiliary regression.

Hypothesis Tests

1. **Null Hypothesis** (H_0): There is no autocorrelation in the residuals of the regression model.
2. **Alternative Hypothesis** (H_1): There is autocorrelation in the residuals of the regression model.

The test statistic used for this hypothesis test is typically the LM (Lagrange Multiplier) statistic, which follows a chi-squared distribution under the null hypothesis.

Interpretation

To interpret the results of the Breusch-Godfrey test:

- If the p-value associated with the LM statistic is less than your chosen significance level (e.g., 0.05), you reject the null hypothesis (H_0). This suggests that there is evidence of autocorrelation in the residuals, indicating that the OLS regression model may not be appropriate.
- If the p-value is greater than your chosen significance level, you fail to reject the null hypothesis. This suggests that there is no strong evidence of autocorrelation in the residuals, and the OLS regression model may be valid.

In our case, we obtain the following results:
The BTC model has an LM statistic of 26.64, which corresponds to a p-value of 0.37. As such, we fail to reject the null hypothesis at the 5% level of significance. This suggests we do not have strong evidence of autocorrelation in the residuals.
The ETH model has an LM statistic of 16.59, which corresponds to a p-value of 0.83. We again fail to reject the null hypothesis at the 5% level of significance. This too suggests we do not have strong evidence of autocorrelation in the residuals.

Durbin-Watson test

Another test we can use is known as the Durbin-Watson test. The Durbin-Watson test is a statistical test used to detect the presence of serial correlation or autocorrelation in the residuals of a regression model (Durbin and Watson (1951)). The Durbin-Watson test provides a test statistic that helps assess whether there is evidence of positive or negative serial correlation in the residuals. Note that this test is less general than the Breusch-Godfrey test above, and is only valid for nonstochastic regressors and for testing the possibility that consecutive errors are related to each other. Nonetheless, here is how the test statistic is calculated:

$$d = \frac{\sum_{i=2}^{n}(\hat{\varepsilon}_i - \hat{\varepsilon}_{i-1})^2}{\sum_{i=1}^{n}\hat{\varepsilon}_i^2}$$

Where:

- $\hat{\varepsilon}_i$ represents the residuals at time i.
- n is the total number of observations.

The Durbin-Watson test statistic, d, takes values between 0 and 4. The interpretation of the test statistic is as follows:
d_l **(Lower Critical Value):** The lower critical value is the smallest Durbin-Watson statistic that would be expected if there were no autocorrelation in the residuals.
d_u **(Upper Critical Value):** The upper critical value is the largest Durbin-Watson statistic that would be expected if there were no autocorrelation in the residuals.

The Durbin-Watson test involves comparing the computed Durbin-Watson statistic to these critical values. Here's what the results typically indicate:

- If the computed Durbin-Watson statistic falls significantly below the lower critical value (close to 0), it suggests positive autocorrelation in the residuals.
- If the computed Durbin-Watson statistic falls significantly above the upper critical value (close to 4), it suggests negative autocorrelation in the residuals.
- If the computed Durbin-Watson statistic is approximately in the middle of the range between the lower and upper critical values (around 2), it suggests that there is no significant autocorrelation in the residuals.

For both the BTC and ETH models, we obtain a Durbin-Watson test statistic of 1.985 and 2.052 respectively. These values are not far enough from 2 to indicate significant evidence of serial correlation in the residuals, given we have D_l and D_u of 1.52 and 1.56 respectively.
Given the lack of evidence for serial correlation of residuals in the two models, we will move on to checking for heteroskedasticity.

2.3.3 Heteroskedasticity

It is assumed that the variance of errors in our models is constant, which is known as the *assumption of homoskedasticity* (Brooks (2008)).
To detect heteroskedasticity, one could examine a graph, such as the plots of residuals against predicted values in Figure 4 and Figure 5. However, given that we are unaware of the cause or form which the heteroskedasticity takes, these plots are not likely to give us much information. As such, we will examine a popular test for heteroskedasticity, the White test.

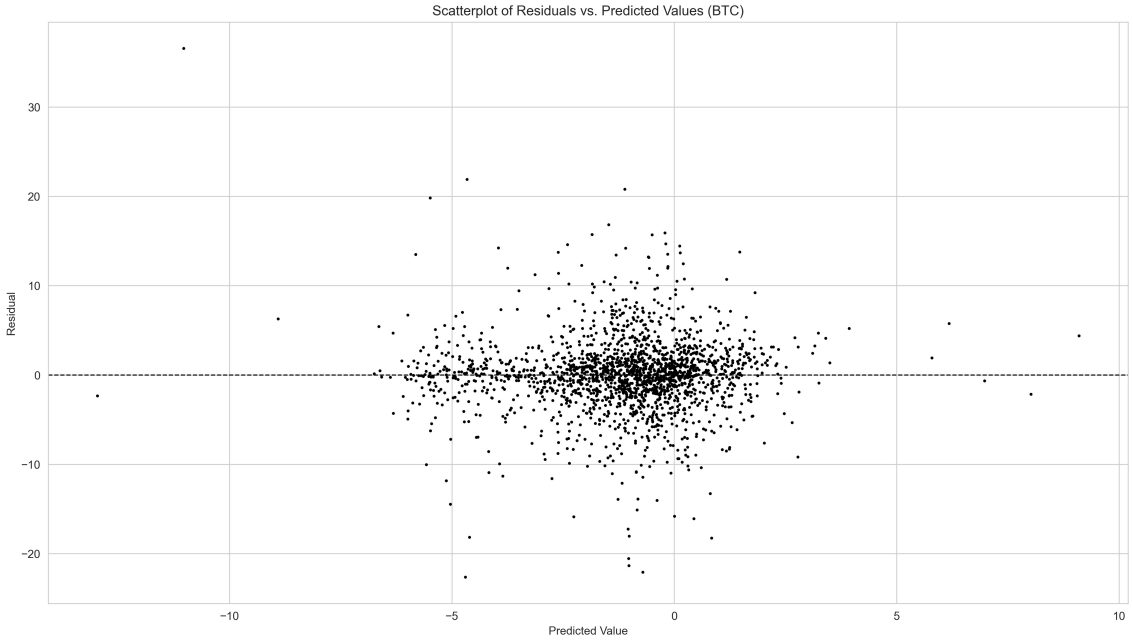


Figure 4: Residuals and Predicted Values - BTC model

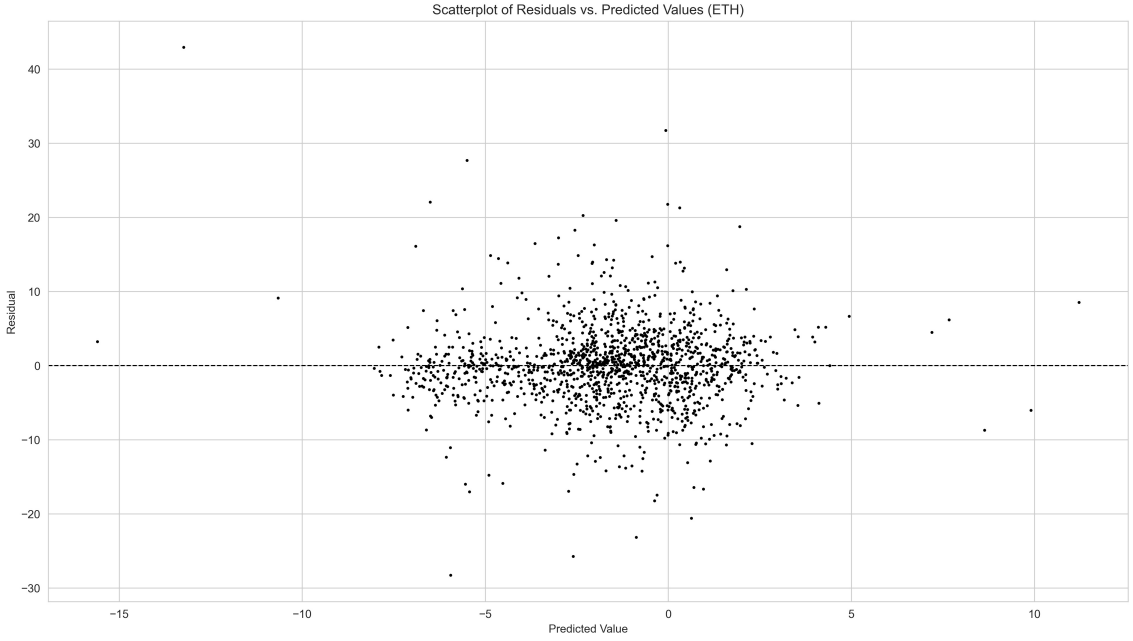


Figure 5: Residuals and Predicted Values - ETH model

White Test

This test is particularly useful because it makes few assumptions about the likely form of the heteroskedasticity. The White test involves two hypotheses:

- **Null Hypothesis (H_0)**: The variance of residuals is constant across different levels of the independent variable(s)(homoskedasticity).
- **Alternative Hypothesis (H_1)**: The variance of residuals is not constant across different levels of the independent variable(s)(heteroskedasticity).

The White test is performed as follows:

1. **Obtain Residuals**: Calculate the residuals from the relevant regression model. Residuals are the differences between the actual observed values and the predicted values from the regression:

$$\hat{\varepsilon}_i = Y_i - \hat{Y}_i$$

Where:

- $\hat{\varepsilon}_i$ is the i-th residual.
- Y_i is the actual observed value.
- $\hat{Y}_i$ is the predicted value from the regression.

2. **Auxiliary Regression**: In this step, you perform an auxiliary regression. You regress the squared residuals ($\hat{\varepsilon}_i{}^2$) from the original regression model onto a set of regressors that contain the original independent variable(s) (X_i) along with their squares(X_i^2) and cross-products. This auxiliary regression helps capture any systematic relationship between the squared residuals and the independent variables:

$$\hat{\varepsilon}_i{}^2 = \gamma_0 + \gamma_1 X_{1i} + \gamma_2 X_{2i} + \ldots + \gamma_k X_{ki} + \ldots$$

Where:

- $\hat{\varepsilon}_i{}^2$ is the squared residual.
- $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$ are the coefficients to be estimated.

3. **Calculate R^2**: After estimating the coefficients in the auxiliary regression, calculate the R^2 value for this regression. This $R^2_{\varepsilon^2}$ represents the proportion of the variation in the squared residuals that is explained by the independent variables in the auxiliary regression.

4. **Lagrange Multiplier Test Statistic**: The White test statistic is calculated as the product of the $R^2_{\varepsilon^2}$ value obtained in the auxiliary regression and the sample size (N). This test

statistic follows a chi-squared distribution with degrees of freedom equal to the number of estimated parameters (P) in the auxiliary regression, excluding the constant term:

$$LM = N \cdot R^2 \sim \chi^2(P-1)$$

5. **Hypothesis Testing**: You then compare the calculated LM test statistic to the critical value from the chi-squared distribution with $(P-1)$ degrees of freedom. If the test statistic is greater than the critical value, you reject the null hypothesis, indicating the presence of heteroskedasticity in the original regression model. If the test statistic is smaller, you fail to reject the null hypothesis, suggesting homoskedasticity.

To interpret the White test:

- If the test statistic LM is significantly greater than the critical value at a chosen significance level, it suggests the presence of heteroskedasticity (rejecting H_0).
- If LM is not significantly greater than the critical value, it implies that there is no strong evidence of heteroskedasticity, and the assumption of homoskedasticity is supported (not rejecting H_0).

In our case, we obtain test statistics of 49.512 and 41.258 for our BTC and ETH models respectively. These correspond to very low p-values, close to zero. As such, for both models, we can reject H_0, and this suggests the presence of heteroskedasticity, which means the residuals in our model do not have constant variance. This violates the assumptions of the model which we have used. Since we have detected heteroskedasticity with the White test, we can use the Breusch-Pagan test for more thorough analysis of the heteroskedasticity.

Breusch-Pagan Test
Again, in the Breusch-Pagan test, we obtain a very low p-value for both models, and thus we reject the null hypothesis for both the BTC model and the ETH model.

For both of our models, we have found heteroskedasticity in the residuals, a violation of the Gauss-Markov assumptions outlined in Section 2.0.1.

To handle the heteroskedasticity in the residuals, we should use robust standard errors and run the linear regression model again. This brings us to the next section of this assignment.

3 Linear Regression with Robust Standard Errors

In Section 2.3 we identified heteroskedasticity in our residuals. This violation of the homoskedasticity assumption can have significant consequences for linear regression analysis:

- Biased Standard Errors: Traditional least squares regression assumes constant variance, resulting in standard errors that may be underestimated or overestimated in the presence of heteroskedasticity.
- Inefficient Estimators: Biased standard errors can lead to inefficient parameter estimators, affecting the precision and accuracy of coefficient estimates.
- Incorrect Inference: heteroskedasticity can lead to incorrect hypothesis tests and confidence intervals, potentially yielding false conclusions about the significance of predictors.

Although we did not find the residuals to be serially correlated, to be on the conservative side and to not make the assumption of no serial correlation (especially as we are using time series data), we will opt to use standard errors that are robust to both heteroskedasticity and serial correlation.

The heteroskedasticity and Autocorrelation Robust Covariance Matrix, commonly referred to as the Newey-West (Newey & West, 1987) estimator, is a statistical tool to address two key issues often encountered in regression analysis: heteroskedasticity and serial correlation (autocorrelation) of residuals.

$$\text{Var}(\hat{\beta}) = \left(X'X\right)^{-1} X'\Omega X \left(X'X\right)^{-1} \tag{11}$$

Where:

- $\hat{\beta}$ represents the vector of estimated coefficients.
- X is the design matrix of the independent variables.
- Ω is the Newey-West estimator matrix that accounts for both heteroskedasticity and serial correlation.

The Newey-West estimator Ω involves a weighted sum of the lagged cross-products of the residuals, with the weights determined by the choice of a kernel function and a bandwidth parameter. The kernel function specifies the weights assigned to different lags of the residuals, and the bandwidth parameter controls the range of lags to consider. By default, the statsmodel package in Python uses Bartlett weights for the kernel weights (statsmodels, 2023).

In summary, the Newey-West estimator is a powerful tool that provides a corrected covariance matrix for regression coefficients. It effectively addresses the issues of heteroskedasticity and serial correlation, making it valuable in time-series analysis, where these issues are common. As such, we can confidently perform statistical inference even when these problems are present in the data, ensuring the validity and reliability of the findings.

Now, we can run our linear regression model again, using robust standard errors as described above. See Table 4 and Table 5 for results.

3.1 Standard Errors

As should be expected, what has changed is that we see slightly larger standard errors in our new models. In the BTC model, the standard error of the intercept coefficient (α) increased from 0.119 to 0.124, and the standard error of the slope coefficient (β) increased from 0.053 to 0.068. For the ETH model, the standard error of the intercept coefficient increased from 0.184 to 0.186, and the standard error of the slope coefficient increased from 0.071 to 0.097. These increases reflect the model's accounting for potential violations of the classical linear regression assumptions.

3.2 Hypothesis Tests

Table 4: Hypothesis Test Results for Robust SE Coefficient Significance (BTC, $\alpha = 0.05$)

Coefficient	**Empirical z**	**Critical z**	**P-Value Test**	**Conf. Interval**	**Result**
$\hat{\alpha}$	1.354	1.96	0.176	[-0.075, 0.412]	fail to reject H_0
$\hat{\beta}$	14.902	1.96	0.000	[0.877, 1.142]	reject H_0

Table 5: Hypothesis Test Results for Robust SE Coefficient Significance (ETH, $\alpha = 0.05$)

Coefficient	**Empirical z**	**Critical z**	**P-Value Test**	**Conf. Interval**	**Result**
$\hat{\alpha}$	1.971	1.96	0.049	[0.002, 0.733]	fail to reject H_0
$\hat{\beta}$	12.679	1.96	0.000	[1.037, 1.416]	reject H_0

For the BTC model, we reject the $\hat{\alpha}$ coefficient. The $\hat{\beta}$ estimate has retained the same level of statistical significance as previously (significant as 1% level). For the ETH model, it is interesting to note that the estimate of the intercept, $\hat{\alpha}$, is now just barely still statistically significant at the 5% level of significance, as in the new model we have $P > z = 0.0487$. The estimate of the slope remains significance the same level of statistical significance (1%).

3.3 Confidence Interval for β

For each cryptocurrency, we will construct a 95% confidence interval for β. This will allow us to test the null hypothesis that the β (asset's risk) is the same as the average risk over the entire market, i.e.

- **Null Hypothesis** (H_0): $\beta = 1$
- **Alternative Hypothesis** (H_1): $\beta \neq 1$

Table 6: Confidence Intervals for Beta

	$\hat{\beta}$	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**	**Contains 1**
BTC	1.0094	0.068	0.139	0.890	0.877	1.142	Yes
ETH	1.2264	0.097	2.341	0.019	1.037	1.416	No

For the BTC model, the value 1 lies within the 95% confidence interval, and as such we fail to reject the null hypothesis that $\beta = 1$.

This means we cannot conclude that Bitcoin has systematic risk which is any more or less risky than the risk the market experiences. For the ETH model, the value 1 does not lie within the 95% confidence interval, and as such we reject the null hypothesis that $\beta = 1$, in favour of the alternative hypothesis, that $\beta \neq 1$. In the case of Ethereum, it appears that it faces more systematic risk than the market.

The above results initially did not match my beliefs about risky and safe returns. Before conducting this test, I expected to find that both cryptocurrencies exhibited returns that were riskier than the market, i.e. $\beta > 1$. Finding that I could not prove β to be different from 1 for Bitcoin, i.e. risk that is different to the market, seemed misleading to me. This is because I knew that Bitcoin exhibited higher volatility than the market (S&P 500) over the period (see Figure 8). However, once I saw the above results I explored this more. Since the β in our models is quantifying how the asset's returns move in relation to the overall market (systematic risk), and the standard deviation (total risk) represents the overall risk of an asset, the standard deviation of the cryptocurrency can be higher while still having $\beta = 1$, since the standard deviation of the cryptocurrency's excess returns will include both systematic and unsystematic risks. Having a CAPM beta of 1 but greater volatility than the market highlights a potential limitation of the CAPM framework, which I will discuss more in the conclusion of the paper, see Section 6.0.2.

4 Estimating CAPM on Subsamples

Next, we will look at taking two smaller subsamples of both the Bitcoin and Ethereum data, so that we can examine if the coefficients estimated for the CAPM models are stable

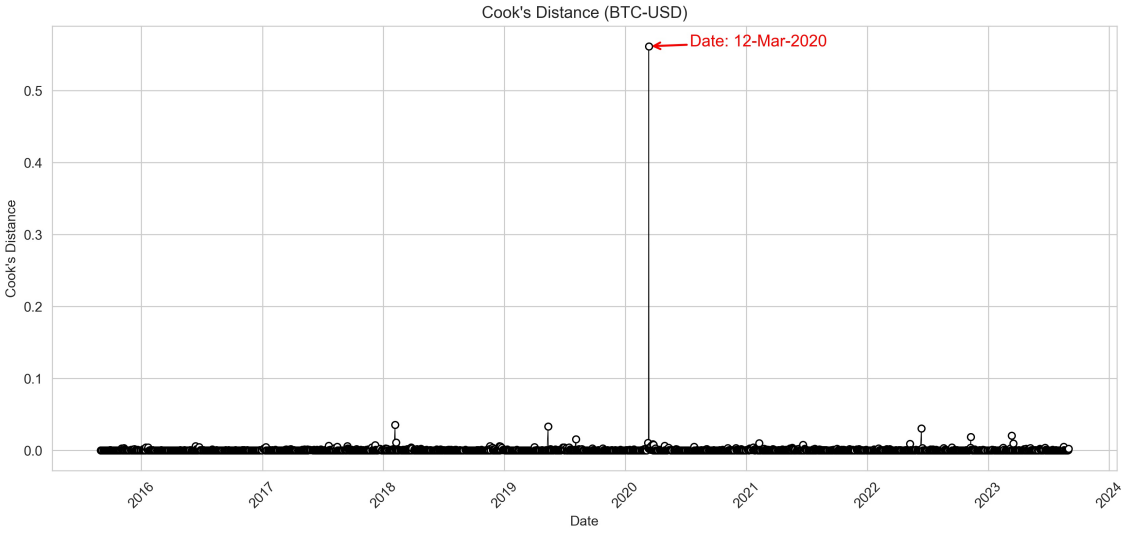


Figure 6: Model Influence - BTC

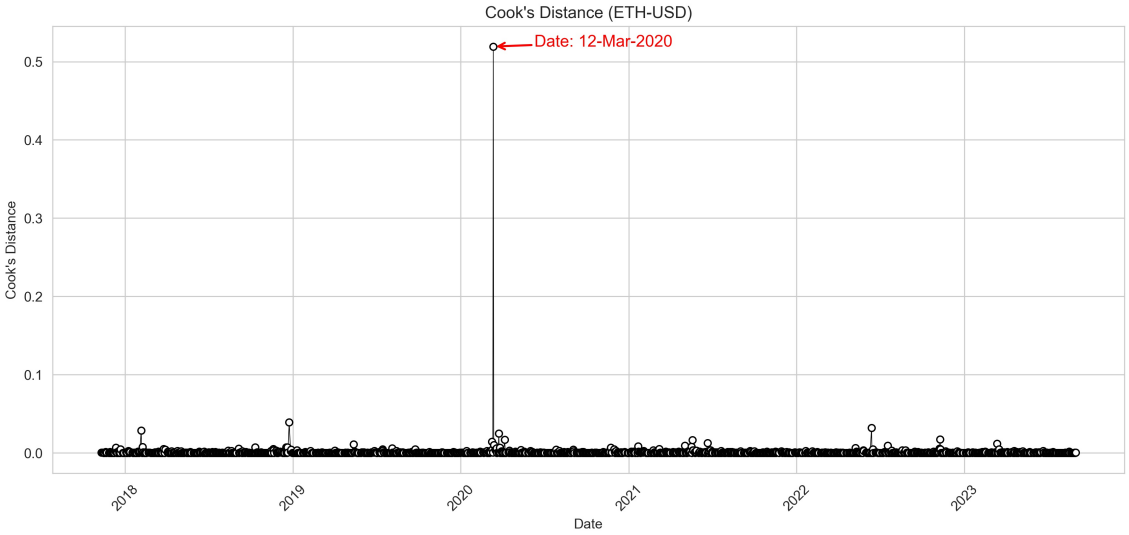


Figure 7: Model Influence - ETH

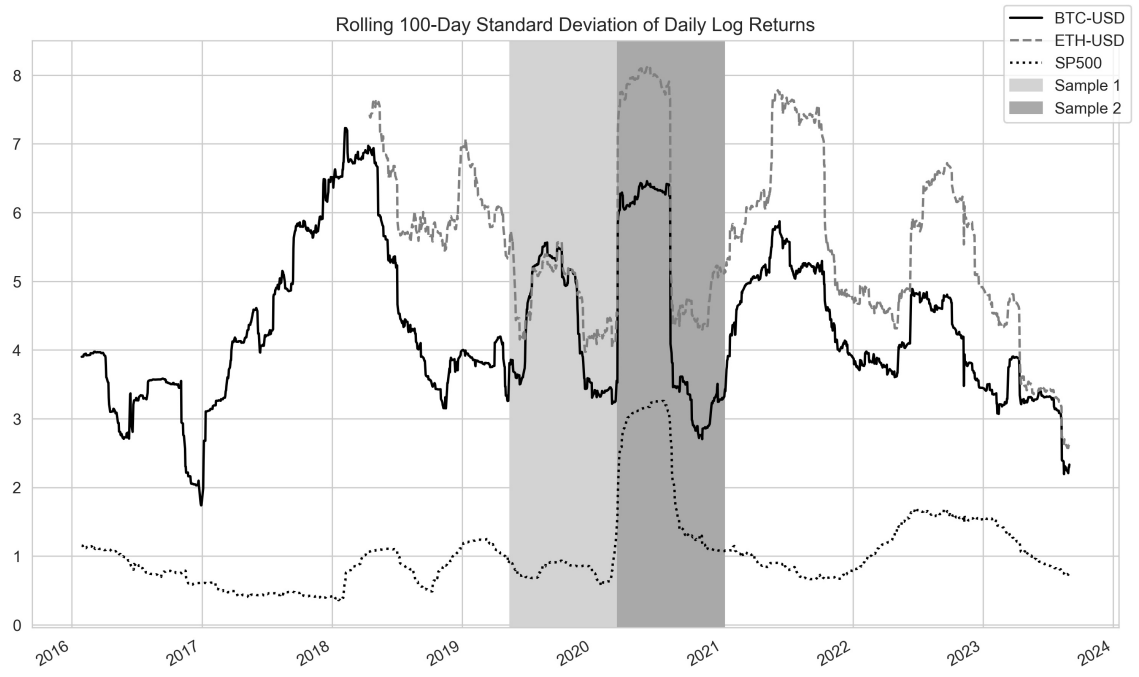


Figure 8: Rolling 100-Day Standard Deviation of Daily Log Returns

over time, or if they vary over time and in different market conditions. I will try to determine whether or not the unusual market conditions and high volatility around the March 2020 period (Figure 8) represents a structural break point in the dataset and if this has an impact on the validity of the CAPM model for pricing cryptocurrency assets. The 16th March 2020 is identified as a key point here, as it represents the largest down day for the S&P 500 in our dataset. Also, the 12th March 2020 is important as this was the worst daily return for both Bitcoin and Ethereum in the dataset. News from that time discusses how Bitcoin lost half its value in a two day period, and Ethereum lost 46% of its value. It is not unreasonable to assume this would have a disproportionate influence on the models we have estimated. To examine which observations contribute the most to the models we have used thus far, we can use Cook's distance (Cook, 1977 & Cook, 1979), denoted as D_i, which allows us to identify which observations in the least squares regression model have the highest influence on the model. Cook's distance for observation i is calculated as:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{Y}_j - \hat{Y}_{j(i)})^2}{p \cdot \text{MSE}}$$

Where:

- $\hat{Y}_j$ represents the predicted value for the jth observation.
- $\hat{Y}_{j(i)}$ represents the predicted value for the jth observation when observation i is excluded.

- p is the number of model parameters (including the intercept).
- MSE is the Mean Squared Error of the model.

A small Cook's distance suggests low influence, while a large value indicates significant influence on the model. For both the Bitcoin and Ethereum models, the 12th March 2020 had by far the highest Cook's distance, and therefore, the highest influence on the models. This is shown in Figures 6 and 7. We will take one sample as the 200 observations previous to, and not including the 12th March 2020, and then the second sample as the 200 observations following and including 12th March 2020. We could expect that our model fit will change anyway, even without a structural change, as now we have much less observations and thus our objective function will look different:

$$\min J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{12}$$

Where:

$J(\theta)$ is the cost function to minimize.

θ are the model parameters (coefficients).

n is the number of data points in the training dataset.

$\hat{y}_\theta(x^{(i)})$ is the predicted value for the i-th data point.

$y^{(i)}$ is the actual target (output) value for the i-th data point.

$\sum_{i=1}^{n}$ denotes the summation over all data points in the training dataset.

For the rest of this section, "Sample 1" refers to the 200 trading day period from 15th May 2019 till the 11th March 2020, and "Sample 2" refers to the 200-day period from 12th March 2020 till the 5th January 2021.
We will begin by going over some of the diagnostic tests from Section 2 for both subsample models, and then compare the two models using the Chow test(Chow, 1960 & Toyoda, 1974).

4.1 Bitcoin Subsamples

We first run a simple linear regression for each sample, as we did in Section 2.

We obtain the following results:

Table 7: Linear Regression Results (BTC Subsamples)

	Sample 1 (A)		Sample 2 (B)	
	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\alpha}$	$\hat{\beta}$
coefficient	-1.5399^{***}	0.2062	0.5475^{*}	1.1195^{***}
$\hat{SE}$	0.552	0.239	0.312	0.142
R^2	0.004		0.239	
Adj. R^2	-0.001		0.235	
F-stat	0.7418		62.21	
Prob(F-stat)	0.390		2.04e-13	
Log-likelihood	-577.16		-579.46	
AIC	1158		1163	
BIC	1165		1170	
Omnibus	18.843		144.319	
Prob(Omnibus)	0.000		0.000	
Durbin-Watson	1.903		1.717	
Jarque-Bera (JB)	57.564		4161.864	
Prob(JB)	3.16e-13		0.00	
Skew	-0.256		-2.240	
Kurtosis	5.578		24.894	

Note: $^{*}\ p < 0.1$; $^{**}\ p < 0.05$; $^{***}\ p < 0.01$

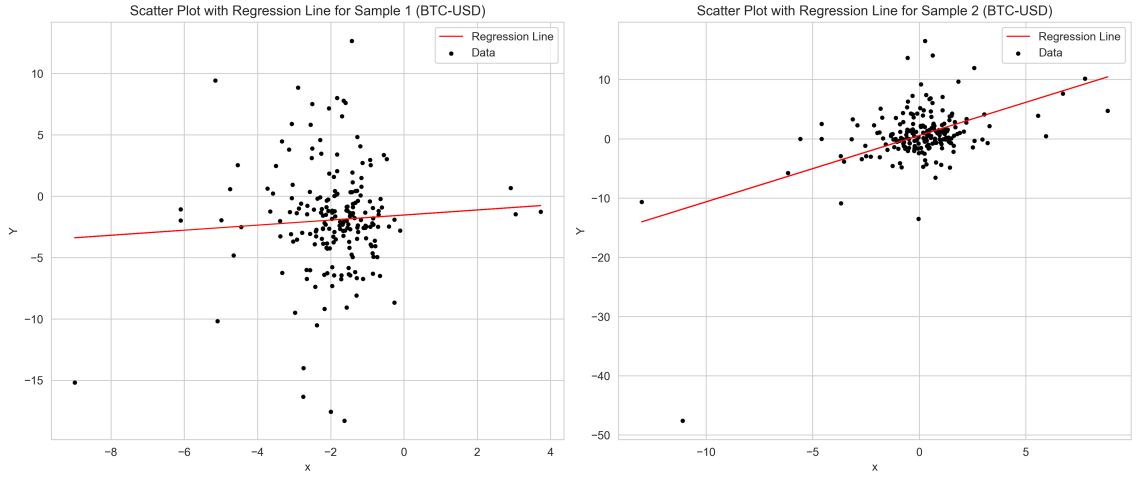


Figure 9: Scatterplots of Regressions on Subsamples (BTC)

4.1.1 Significance of Coefficients

$$\begin{cases} H_0 : \beta_i = 0 \\ H_1 : \beta_i \neq 0 \end{cases}$$

Table 8: Hypothesis Test Results for Coefficient Significance - BTC (Sample 1, $\alpha = 0.05$)

Coefficient	Empirical T	Critical T	P-Value Test	Conf. Interval	Result
$\hat{\alpha}$	-2.792	1.96	0.006	[-2.628, -0.452]	reject H_0
$\hat{\beta}$	0.861	1.96	0.390	[-0.266, 0.678]	fail to reject H_0

Table 9: Hypothesis Test Results for Coefficient Significance - BTC (Sample 2, $\alpha = 0.05$)

Coefficient	Empirical T	Critical T	P-Value Test	Conf. Interval	Result
$\hat{\alpha}$	1.756	1.96	0.081	[-0.067, 1.162]	fail to reject H_0
$\hat{\beta}$	7.887	1.96	0.000	[0.840, 1.399]	reject H_0

4.1.2 Statistical Indicators and Results

Coefficient Estimates

For the model obtained from sample 1 (henceforth Model A), we get $\hat{\alpha}$ and $\hat{\beta}$ coefficient estimates of -1.5399 and 0.2062 respectively. The $\hat{\alpha}$ estimate is statistically significant at the 1% level of significance. Comparing this to the model obtained from sample 2 (henceforth Model B), we obtained $\hat{\alpha}$ and $\hat{\beta}$ estimates of 0.5475 and 1.1195 respectively. The $\hat{\alpha}$ estimate was significant at the 10% level of significance, while the $\hat{\beta}$ estimate was significant at the 1% level of significance. In the first period, we estimated that a 1% increase in the excess return of the market corresponds to a 1.5399% decrease in the excess return of Bitcoin. For the second period, we find that a 1% increase in the excess return of the market corresponds to a 0.5475% increase in the excess return of the Bitcoin. There is clearly quite a difference between the models in the two different sample periods, and only the β estimate in the second sample is significant.

R-squared

For model A, we had an R^2 value of 0.004. This is very close to zero and indicates that the excess return of the market explains almost none of the variation of the excess return of Bitcoin in this sample period, which is a very poor fit.

For model B, we had an R^2 value of 0.239. This is higher and indicates a better fit during the second period, but although slightly better, still indicates that just 23.9% of the variation of excess return of Bitcoin can be explained by the excess return of the market during sample 2's period.

F-statistic

For model A, an F-statistic of 0.7418 is obtained. We have an f-statistic of 62.21 for model B. We can use the F-table to find the critical value for significance level $\alpha = 0.05$ and $\mathscr{F}(2-1, 200-2)$. We then obtain a critical f-value of 3.89. For model A, we have a p-value of 0.390 for our f-stat, and therefore fail to reject the null. For model B, the f-stat corresponds to a p-value of 2.04e-13, which is very small, and as such, we can reject the

null hypothesis and conclude that at least one of the coefficient estimates is significant.
To conclude, we find that the second sample provides a more significant fit than the first sample.

Omnibus
Model A has an omnibus value of 18.843, while model B has a value of 144.319. These both correspond to p-values very close to zero, and indicates residuals are likely not normally distributed for both models.

Jarque-Bera
For the BTC model, we obtain a Jarque-Bera statistic of 3384.188, which corresponds to a p-value of 0.00. Therefore, we find that the residuals deviate significantly from a normal distribution.
For the ETH model, we obtain a Jarque-Bera statistic of 2226.082, which corresponds to a p-value of 0.00. Again, we find that the residuals deviate significantly from a normal distribution, although perhaps to a lesser degree than the BTC model.

Skewness
For model A, we have a skew figure of -0.256, while the model B has a skew figure of -2.240. Given the large negative day(s) seen in the second sample, it makes sense to see it is more negatively skewed than the other sample.

Kurtosis
For model A, we have kurtosis figure of 5.578, while model B has kurtosis of 24.894. Again, kurtosis in residuals can reveal information about the tails of the distribution, and, specifically, these leptokurtic residuals may indicate the presence of outliers or extreme values that are not accounted for by the model. Again, given the second sample had the most extreme negative day in the entire sample, it makes sense to see it has a higher kurtosis.

4.1.3 Chow Test

The Chow test is a statistical test used to determine whether there is a significant difference in the coefficients of two separate regressions. It can be used to assess the presence of structural breaks or changes in the relationship between variables at a specific point in time.

The Chow test is based on the following equation:

$$teststatistic = \frac{RSS - (RSS_1 + RSS_2)}{RSS_1 + RSS_2} \times \frac{n - 2p}{p}$$

Where:

RSS : Sum of squared residuals for the entire dataset (both samples)

RSS_1 : Sum of squared residuals for the first sample

RSS_2 : Sum of squared residuals for the second sample

p : Number of model parameters (regression coefficients) for each sample

n : Total number of observations

Interpretation

The Chow test (Chow, 1960 and Toyoda, 1974) assesses whether there is a significant structural break between the two segments. The test statistic F is compared to a critical value from the F-distribution with appropriate degrees of freedom. If F is greater than the critical value, it suggests that there is a significant difference in the coefficients between the segments, indicating a structural break.

In our case, we obtain a chow statistic of 6.647. This is above the critical value of 3.019, and corresponds to a p-value of 0.001. As such, we find that there is a significant difference in the coefficients between the samples (at the 1% level of significance), which indicates a structural break before and after the market volatility of March 2020.

4.2 ETH Subsamples

Next, we run the same simple linear regression for each sample for the ETH dataset, similar to in Section 2 but with the different subsamples.

We obtain the following results

Table 10: Linear Regression Results (ETH Subsamples)

	Sample 1		Sample 2	
	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\alpha}$	$\hat{\beta}$
coefficient	-0.9255	0.5539^{**}	0.5611	1.5121^{***}
$\hat{SE}$	0.623	0.270	0.399	0.181
R^2	0.021		0.260	
Adj. R^2	0.016		0.256	
F-stat	4.203		69.47	
Prob(F-stat)	0.0417		1.29e-14	
Log-likelihood	-601.39		-628.54	
AIC	1207		1261	
BIC	1213		1268	
Omnibus	20.332		96.525	
Prob(Omnibus)	0.000		0.000	
Durbin-Watson	1.844		1.919	
Jarque-Bera (JB)	39.026		1347.931	
Prob(JB)	3.35e-09		2.00e-293	
Skew	-0.505		-1.425	
Kurtosis	4.914		15.395	

Note: $^{*}\,p < 0.1$; $^{**}\,p < 0.05$; $^{***}\,p < 0.01$

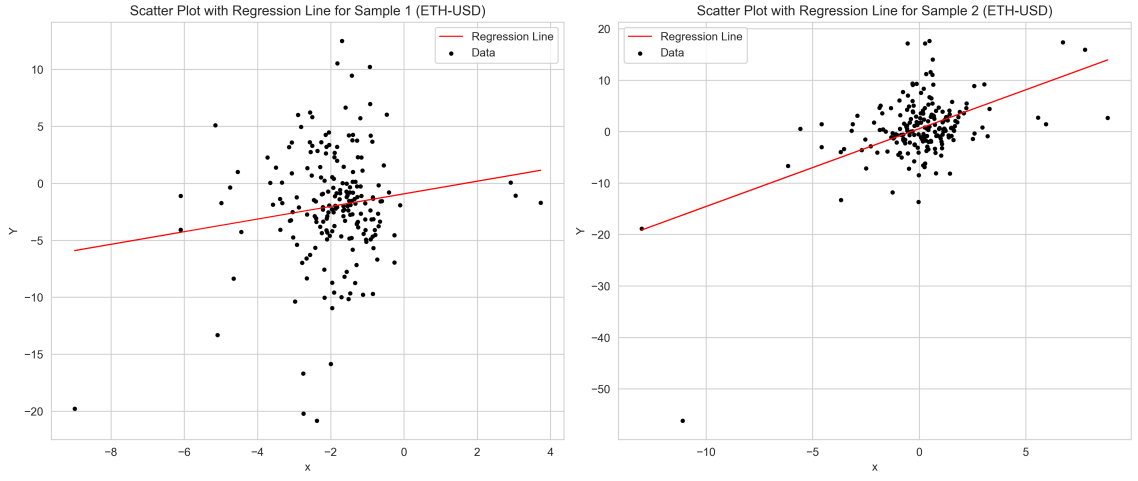


Figure 10: Scatterplots of Regressions on Subsamples (ETH)

4.2.1 Significance of Coefficients

$$\begin{cases} H_0 : \beta_i = 0 \\ H_1 : \beta_i \neq 0 \end{cases}$$

Table 11: Hypothesis Test Results for Coefficient Significance: ETH (Sample 1, $\alpha = 0.05$)

Coefficient	Empirical T	Critical T	P-Value Test	Conf. Interval	Result
$\hat{\alpha}$	-1.487	1.96	0.139	[-2.153, 0.302]	fail to reject H_0
$\hat{\beta}$	2.050	1.96	0.042	[0.021, 1.087]	reject H_0

Table 12: Hypothesis Test Results for Coefficient Significance: ETH (Sample 2, $\alpha = 0.05$)

Coefficient	Empirical T	Critical T	P-Value Test	Conf. Interval	Result
$\hat{\alpha}$	1.408	1.96	0.161	[-0.225, 1.347]	fail to reject H_0
$\hat{\beta}$	8.335	1.96	0.000	[1.154, 1.870]	reject H_0

4.2.2 Statistical Indicators and Results

Coefficient Estimates

For the model obtained from sample 1 (henceforth Model A), we get $\hat{\alpha}$ and $\hat{\beta}$ coefficient estimates of -0.9255 and 0.5539 respectively. The $\hat{\beta}$ estimate is statistically significant at the 5% level of significance. Comparing this to the model obtained from sample 2 (henceforth Model B), we obtained $\hat{\alpha}$ and $\hat{\beta}$ estimates of 0.5611 and 1.5121 respectively. The $\hat{\beta}$ estimate was significant at the 1% level of significance. Both models have β coefficient estimates which are significant at the 5% level of significance, while both $\hat{\alpha}$ coefficient estimates are insignificant. In the first period, we estimated that a 1% increase in the excess return of the market corresponds to a 0.5539% decrease in the excess return of Ethereum. For the second period, we find that a 1% increase in the excess return of the market corresponds to a 1.5121% increase in the excess return of the Ethereum. Although both positive, we find there to be a big difference in the magnitude of the relationship between $(R_{ETH} - R_f)$ and $(R_m - R_f)$ when comparing the first period to the second period. As such there looks to be a structural change at the break point we identified.

R-squared

For model A, we had an R^2 value of 0.021. This is very close to zero, and indicates that the excess return on the market explains almost none of the variation of the excess return of Ethereum in this sample period, which is a very poor fit.

For the B, we had an R^2 value of 0.260. This is higher and indicates a better fit, but although slightly better, still indicates that just 26% of the variation of excess return of Ethereum can be explained by the excess return of the market during sample 2's period. This indicates that in the second sample, Ethereum seems to move more in line with the market than in the first sample.

F-statistic

For model A, an F-statistic of 4.203 is obtained. We have an f-statistic of 69.47 for model B. We can use the F-table to find the critical value for significance level $\alpha = 0.05$ and $\mathscr{F}(2-1, 200-2)$. We then obtain a critical f-value of 3.89. For model A, we have a

p-value of 0.0417 for our f-stat, so we can reject the null hypothesis at the 5% level of significance. For model B, the f-stat of 69.47 corresponds to a p-value of 1.29e-14, which is very small, and as such, we can reject the null hypothesis and conclude that at least one of the coefficient estimates is significant.
To conclude, we find that models for both samples are statistically significant, but the second sample provides a more statistically significant fit than the first sample.

Omnibus
Model A has an omnibus value of 20.332, while model B has a value of 96.525. These both correspond to p-values very close to zero, and indicates residuals are likely not normally distributed for both models.

Jarque-Bera
For model A, we obtain a Jarque-Bera statistic of 39.026, which corresponds to a p-value of very close to zero. Therefore, we find that the residuals deviate significantly from a normal distribution.
For model, we obtain a Jarque-Bera statistic of 1347.931, which also corresponds to a p-value of very close to zero. Again, we find that the residuals deviate significantly from a normal distribution.

Skewness
For model A, we have a skew figure of -0.505, while the model B has a skew figure of -1.425. Both models have a tail on the left side which is longer or fatter. Again, we find the second sample to have residuals which are more negatively skewed.

Kurtosis
For model A, we have kurtosis figure of 4.914, while model B has kurtosis of 15.395. Model B may have more outliers or extreme values which are unaccounted for by the model. Similar to what we found for BTC, this makes sense given the nature of the observations in the second sample.

4.2.3 Chow Test

In this case, we obtain a chow statistic of 4.075. This is above the critical value of 3.019 and corresponds to a p-value of 0.018. As such, we find that there is a significant difference in the coefficients between the samples, which again indicates a structural break at the chosen point. This is at the 5% level of significance.

4.3 Comment

We have found that both Bitcoin and Ethereum, before and after the break point of 12th March 2020, have significantly different coefficient estimates in the CAPM. These coefficient estimates also differ considerably from those that we found in Section 2. While we cannot confirm for certain exactly what causes the structural difference, it highlights a

potential limitation of CAPM, the fact that over time the parameters underlying the model fluctuate. This limitation will be discussed more in the conclusion, see Section 6.

5 Estimating CAPM Without a Constant

Next, we will explore what happens when we run our models without the constant, α. Since the CAPM model assumes α is zero, it will be interesting to see how our models' fits change compared with Section 2. When a simple linear regression is run without a constant (intercept), the regression model is expressed as:

$$Y_i = \beta X_i + \varepsilon_i \tag{13}$$

where:

Y_i represents the dependent variable for the i-th observation,
X_i represents the independent variable (predictor) for the i-th observation,
β is the slope coefficient,
ε_i is the error term for the i-th observation.

In this case, the regression line does not have an intercept term. This means that the line is forced to pass through the origin (0,0), and the model assumes that when the independent variable is zero, the dependent variable is also zero. For CAPM, this means we are assuming $\alpha = 0$, meaning that when the log market excess is return is zero, the log excess return for the crypto currency will be zero also.

It also means that some undesirable consequences may arise (Brooks, 2008). Firstly, the coefficient of determination (R^2), which is calculated as the ratio of explained sum of squares (ESS) to the total sum of squares (TSS), may become negative. This implies that the sample average, $\bar{y}$, seems to "explain" more of the variation in the dependent variable (y) than the explanatory variables themselves. Secondly, and perhaps more significantly, a regression model without an intercept parameter can introduce potentially significant biases in the estimates of the slope coefficients.

Given that in Section 2.2 we found the Bitcoin model did not have a statistically significant intercept, or $\hat{\alpha}$ coefficient, we might expect this model's fit to not change too much following the removal of the intercept. However, for Ethereum, we found the $\hat{\alpha}$ coefficient to be significant at the 5% level of significance, and thus this model fit may dis-improve.

5.0.1 Results

Table 13: Linear Regression Results (no constant)

	BTC	ETH
	$\hat{\beta}$	$\hat{\beta}$
Coefficient	0.9965***	1.1389***
$\hat{SE}$	0.043	0.056
R^2†	0.206	0.226
Adj. R^2†	0.206	0.226
F-stat	504.2	410.7
Prob(F-stat)	1.92e-99	2.67e-80
Log-likelihood	-5579.2	-4375.9
AIC	1.116e+04	8754
BIC	1.116e+04	8759
Omnibus	283.336	253.424
Prob(Omnibus)	0.000	0.000
Durbin-Watson	1.983	2.050
Jarque-Bera (JB)	3521.636	2379.470
Prob(JB)	0.00	0.00
Skew	-0.209	-0.546
Kurtosis	9.586	9.281

Note: * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

†: uncentered

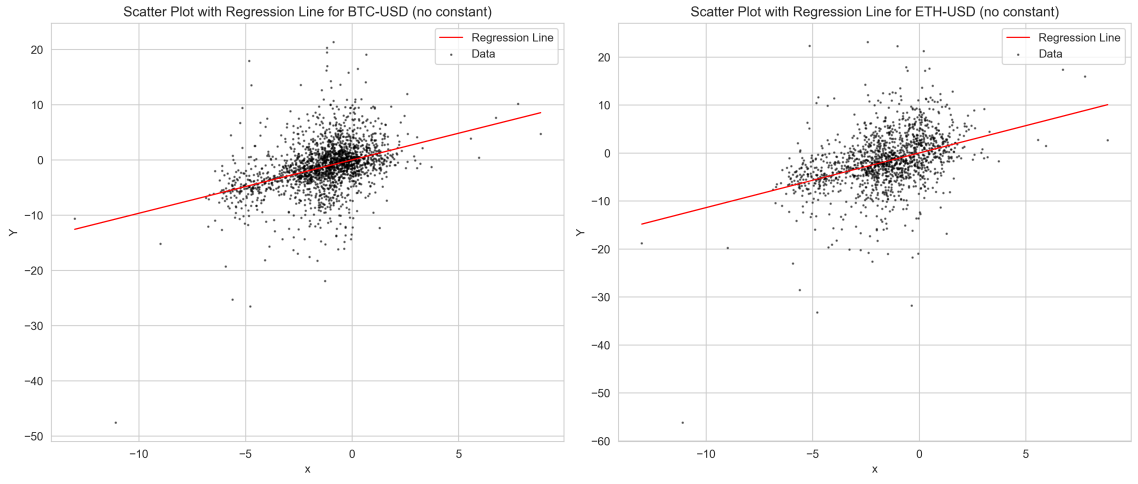


Figure 11: Regression Plot for BTC and ETH (no constant).

5.0.2 Statistical Indicators

Coefficient Estimates

For the BTC model without a constant, we get a $\hat{\beta}$ coefficient estimate of 0.9965. This

estimate is statistically significant at the 1% level of significance. This means for we expect Bitcoin to closely match the returns of the market, given this model has an intercept of zero and a $\hat{\beta}$ of almost 1. For the ETH model, we have $\hat{\beta}$ estimate of 1.1389. This is also significant at the 1% level of significance. This suggests that for a 1% rise (fall) in the excess return of the market, we expect Ethereum to rise (fall) by approximately 1.14%.

R-squared

Note: the R^2 values are usually meaningless in a regression with no intercept, since the mean value of the independent variable, $\bar{y}$, won't be equal to the mean of the fitted values from the model, $\hat{y}$ (Brooks, 2008). As such, we won't use these values to comment on the fit of the model.

F-statistic

For the BTC model, an F-statistic of 504.2 is obtained. We have an f-statistic of 410.7 for the ETH model. We can use the F-table to find the critical value for significance level $\alpha = 0.05$ and $\mathscr{F}(2-1, 1941-1)$. We then obtain a critical f-value of 3.85. For the BTC model, we have a p-value of 1.92e-99 for our f-stat, so we can reject the null hypothesis at the 1% level of significance. For ETH, the f-stat corresponds to a p-value of 2.67e-80, which is very small, and as such, we can again reject the null hypothesis at the 1% level of significance.

For a regression with no intercept and just one independent variable, the F-test is not very meaningful. We concluded that both models are significantly better than just using the intercept, but we could probably infer this from seeing the significance of our estimates in Table 13. We are effectively saying that, since our f-statistic is statistically significant, it means our models' predictions are an improvement over just using the mean (intercept only).

Log Likelihood

We get log likelihood statistics of -5579.2 and -4375.9 for the BTC and ETH models respectively. We can conduct a log likelihood ratio test in order to compare these new models with no intercept to the models we previously had in Section 2

The log-likelihood ratio statistic (G) is calculated as follows:

$$G = -2 \times (\text{log-likelihood of null model} - \text{log-likelihood of alternative model})$$

The LLR statistic follows a chi-squared distribution with degrees of freedom equal to the difference in the number of parameters between the two models. Using the chi-squared distribution and the LLR statistic, we calculate the p-value. A small p-value suggests that the alternative model is a significantly better fit for the data. A large p-value suggests that there is no significant difference between the models, and you should stick with the simpler null model.

For Bitcoin, we fail to reject the null hypothesis at the 5% level of significance, i.e. we

conclude that the null model (from Section 2) is as good a fit as the model with no intercept.
For Ethereum, we have the same result.

Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC)
For BTC, we have AIC and BIC values of 11,160 and 11,165 respectively. We can compare this AIC value to the AIC value for the model in Section 2, which was only very slightly larger. For BIC, we have a slightly lower value in our new model with no intercept, which makes sense as BIC is generally quite parsimonious and will recommend the model with the least number of regressors.
For ETH, we have AIC and BIC values of 8754 and 8759 respectively. Comparing the AIC value to the previous value of 8752, we can see that the model in Section 2 actually had a lower value which indicates it is a better model. The BIC is again lower for the new model.

Omnibus
The BTC model has an omnibus value of 283.336, while model B has a value of 253.424. These both correspond to p-values very close to zero, and indicates residuals are likely not normally distributed for both models.

Jarque-Bera
For the BTC model, we have a Jarque-Bera statistic of 3521.636, which corresponds to a p-value of very close to zero. Therefore, we find that the residuals deviate significantly from a normal distribution.
For the ETH model, we obtain a Jarque-Bera statistic of 2379.470, which also corresponds to a p-value of very close to zero. Again, we find that the residuals deviate significantly from a normal distribution.

Skewness
For BTC, we have a skew figure of -0.209, while the ETH model has a skew figure of -0.546. Both models have a tail on the left side which is longer or fatter.

Kurtosis
For the BTC model, we have kurtosis figure of 9.586, while the ETH model has kurtosis of 9.281. These values are similar and indicate that both models have outliers or extreme values which are unaccounted for by the model.

5.1 Comment

Since in Section 2, we found that in the BTC model the intercept, or α estimate, was not significantly different from zero, we did not find a particularly large difference in the model when removing the intercept. For the ETH model however, we had previously found the α estimate to be significantly different to zero, so there was a bigger impact on

the estimate of β when removing the intercept.
Overall, we did not find the difference in model fit with or without the intercept to be statistically significant.

6 Conclusion

There are many reasons why CAPM is a beneficial and useful framework for modelling risk and returns. In my view, the main benefit of CAPM lies in its simplicity. It provides a straightforward framework for estimating expected returns, which can be valuable when analysing different assets, like cryptocurrencies, and comparing them to the market portfolio. This simplicity stems from the fact that it assumes that only one source of risk, market risk, affects expected returns. Another advantage of CAPM is in its usefulness as a benchmark. For example, we saw throughout this assignment how I could easily compare and contrast the models for both Bitcoin and Ethereum. CAPM makes it easy to evaluate and interpret their performance relative to the market portfolio.
However, there were also issues and limitations associated with CAPM which I encountered throughout this assignment, and which I will expand on here.

6.0.1 Non-Normality of Returns

Firstly, we observed that residuals were not normally distributed in our regression models. CAPM assumes that asset returns follow a normal distribution, and non-normality can lead to biased parameter estimates and affect the validity of statistical tests. In the context of cryptocurrencies, we saw that returns often exhibit heavy tails and significant skewness, which does not align with the normal distribution assumption. CAPM does not handle this feature of cryptocurrencies well.

6.0.2 Beta as Key Driver of Returns

Another potential limitation of CAPM is the assumption that β (systematic risk) is the primary driver of returns. This does not directly account for total risk (which could be measured by standard deviation/volatility), and unsystematic risk which is not explained by the market. While I stated the benefit of this was simplicity, the drawback of this is that it is quite a big simplification and it rarely holds in practice, as our results showed. We conclude from the low R^2 values in our models that excess return of the S&P 500 has very little explanatory power when it comes to the excess returns of cryptocurrencies. In practice, other factors, including unsystematic risk, can heavily influence an asset's volatility. In Section 3.2, I touched on the fact that while the cryptocurrencies had β estimates close to, or slightly above 1, their returns had volatility which was far higher than that of the market. Therefore, while a β of 1 implies systematic risk similar to the market, standard deviation may capture additional factors contributing to the asset's total

risk. This could be slightly misleading, perhaps for a less sophisticated investor, who believes they are investing in an asset which has similar risk to the market but is in fact far more volatile. In such cases, it's essential to consider both market beta and volatility and not rely solely on CAPM for assessing risk and making investment decisions.

6.0.3 Market Portfolio Representation

Additionally, a potential issue with CAPM lies in deciding what to use to represent the market portfolio. In this analysis, I used the S&P 500, but as we saw, this was not a useful proxy to explain much of the variance of the two cryptocurrencies. We highlighted the issue of using the S&P 500 to model cryptocurrency returns, that the return of a market cap weighted basket of US equities cannot capture the risk and unique variance associated with cryptocurrencies. Cryptocurrency risk is clearly influenced by factors that are distinct from traditional equity markets. In this regard, perhaps using some sort of cryptocurrency index would be more representative as the market portfolio for modelling cryptocurrencies. In reality, other sources of risk almost certainly exist in the cryptocurrency market and omitting them from the model can lead to omitted variable bias. Additional risk factors, such as macroeconomic variables or specific crypto-related factors, may need to be considered to improve the fit of the model. Some of these risk factors could include size, momentum, volume and volatility (Liu, Tsyvinski, & Wu, 2022).

6.0.4 Instability of Coefficient Estimates Over Time

Another issue we discovered with the CAPM model was the instability of the coefficient estimates. We found that over time these estimates may change, so the relationship between cryptocurrencies and the market may not be constant. The assumptions of CAPM may not hold throughout time periods with varying market conditions, perhaps such as those in March 2020 that we explored in Section 4.

6.0.5 Closing Thoughts

In conclusion, while CAPM offers a simple and valuable framework for understanding expected returns in traditional financial markets, its direct application to cryptocurrencies comes with challenges and limitations. Cryptocurrencies exhibit unique characteristics, including non-normal returns, high volatility, and potential omitted risk factors. Researchers and investors should be cautious when using CAPM for cryptocurrencies. The appropriateness of CAPM as a tool to understand the dynamics of cryptocurrencies will really depend on a practitioner's ability to consider model adaptations that better account for the idiosyncrasies of the cryptocurrency market. One such adaptation was a simple three-factor pricing model (Shen, Urquhart, & Wang, 2020), consisting of market, size, and reversal factors, which was found to significantly outperform the traditional CAPM model when applied to cryptocurrencies.

References

AKdemy. (2021). *Why and when we should use the log variable.* Retrieved from https://quant.stackexchange.com/questions/64028/why-and-when-we-should-use-the-log-variable (Accessed on October 13, 2023)

Anderson, T. W., & Darling, D. A. (1952, June). Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *Ann. Math. Statist.*, *23*(2), 193–212.

Breusch, T. S. (1978). Testing for autocorrelation in dynamic linear models. *Australian Economic Papers*, *17*(31), 334-355. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8454.1978.tb00635.x doi: 10.1111/j.1467-8454.1978.tb00635.x

Brooks, C. (2008). *Introductory econometrics for finance (second edition)*. Cambridge, UK: Cambridge University Press.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, *28*(3), 591-605. Retrieved from https://www.jstor.org/stable/1910133 doi: 10.2307/1910133

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, *19*(1), 15–18. Retrieved from https://doi.org/10.2307/1268249 doi: 10.2307/1268249

Cook, R. D. (1979). Influential observations in linear regression. *Journal of the American Statistical Association*, *74*(365), 169–174. Retrieved from https://doi.org/10.2307/2286747 doi: 10.2307/2286747

D'Agostino, R. B. (1971). An omnibus test of normality for moderate and large size samples. *Biometrika*, *58*(2), 341–348. Retrieved from https://www.jstor.org/stable/2334522 doi: 10.2307/2334522

D'Agostino, R. B., Belanger, A., & D'Agostino, J., Ralph B. (1990). A suggestion for using powerful and informative tests of normality. *The American Statistician*, *44*(4), 316–321. Retrieved from https://www.jstor.org/stable/2684359 doi: 10.2307/2684359

Durbin, J., & Watson, G. S. (1951). Testing for serial correlation in least squares regression. ii. *Biometrika*, *38*(1/2), 159–77. doi: 10.2307/2332325

Godfrey, L. G. (1978). Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica*, *46*(6), 1293-1301. Retrieved from https://www.jstor.org/stable/1913829 doi: 10.2307/1913829

Liu, Y., Tsyvinski, A., & Wu, X. (2022). Common risk factors in cryptocurrency. *The Journal of Finance*, *77*(2), 1133-1177.

Newey, W. K., & West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, *55*(3), 703-708. Retrieved from https://doi.org/10.2307/1913610 doi: 10.2307/1913610

Shen, D., Urquhart, A., & Wang, P. (2020). A three-factor pricing model for cryptocurrencies. *Finance Research Letters*, *34*, 101248. Retrieved from https://www.sciencedirect.com/science/article/pii/S1544612319304519 doi: 10.1016/j.frl.2019.07.021

statsmodels. (2023). *statsmodels.* https://www.statsmodels.org/stable/generated/statsmodels.stats.sandwich_covariance.cov_hac.html. (Accessed on October 13, 2023)

Toyoda, T. (1974). Use of the chow test under heteroscedasticity. *Econometrica*, *42*(3), 601-608. Retrieved from https://www.jstor.org/stable/1911796 doi: 10.2307/1911796

Appendix

A Additional Charts

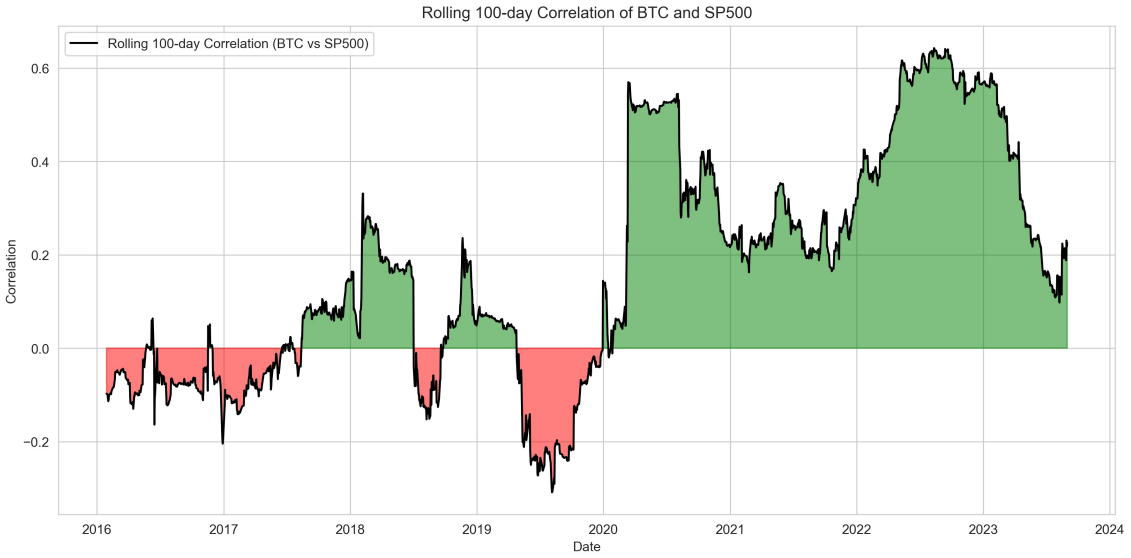


Figure A.1: Rolling 100-Day Correlation of BTC and S&P500

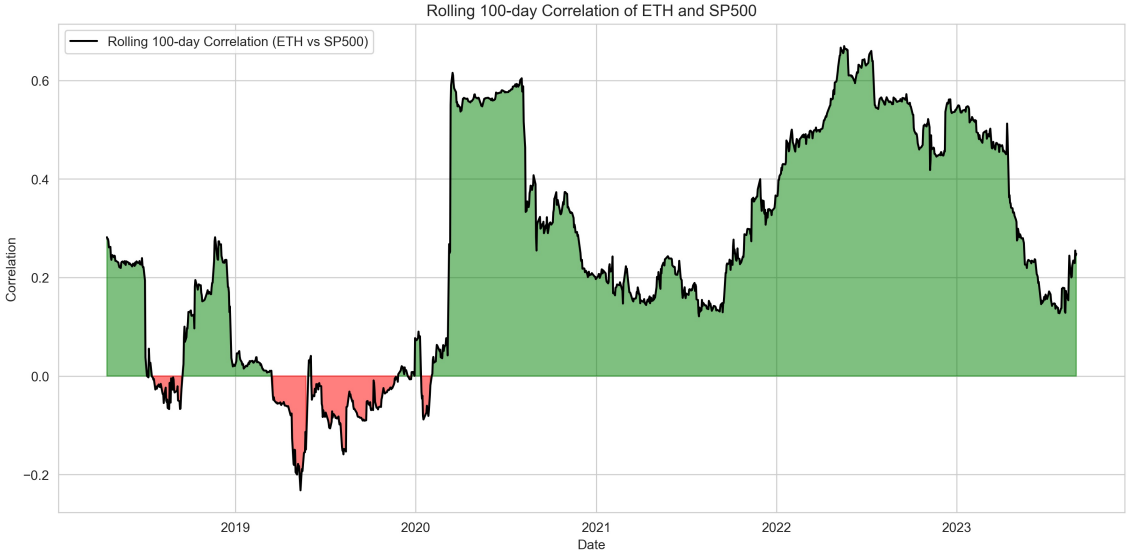


Figure A.2: Rolling 100-Day Correlation of ETH and S&P500

Code

October 15, 2023

1 Importing Libraries

[296]:
```
import yfinance as yf
import pandas as pd
import numpy as np
from datetime import datetime
import statsmodels.api as sm
import matplotlib as mpl
import matplotlib.pyplot as plt
from fredapi import Fred
import scipy.stats as stats
from scipy.stats import f, chi2
import seaborn as sns
from tabulate import tabulate
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import het_white, het_breuschpagan,␣
↪acorr_breusch_godfrey, acorr_ljungbox
```

[6]:
```
import warnings
warnings.filterwarnings("ignore")
```

[130]:
```
sns.set_style('whitegrid')
```

2 Gathering Data

3 a) & b)

a) Using https://finance.yahoo.com/cryptocurrencies, download the prices of 2 cryptocurrencies from September 2015 (or the oldest release) to August 2023 (in his- torical data). Tranform them adequately to have log return x 100 (rt=ln(Pt/Pt−1)x 100). Using the dataset maintained by Federal Reserve of St. Louis https://fred.stlouisfed.org/, download SP500 and risk free interest rate (FFR) for the same sample period.

[8]:
```
cryptos = yf.download("BTC-USD ETH-USD")
cryptos = cryptos['Close']

start = pd.to_datetime('2015-08-31')
```

```
cutoff = pd.to_datetime('2023-08-31')
```

```
[**********100%**********]  2 of 2 completed
```

[9]:
```
BTC = cryptos[['BTC-USD']]
ETH = cryptos[['ETH-USD']]
```

download SP500 and risk free interest rate (FFR)

[10]:
```
fred = Fred(api_key='yourkeyhere')

SP = pd.DataFrame(fred.get_series('SP500'))
SP = SP.rename(columns={0:"SP500"})

FFR = pd.DataFrame(fred.get_series('DFF'))
FFR = FFR.rename(columns={0:"FFR"})
```

[11]:
```
BTC=BTC.merge(SP, left_index=True, right_index=True)
BTC=BTC.merge(FFR, left_index=True, right_index=True)

ETH=ETH.merge(SP, left_index=True, right_index=True)
ETH=ETH.merge(FFR, left_index=True, right_index=True)
```

[12]:
```
def log_returns(df, crypto_name):
    for col in [crypto_name, 'SP500']:
        df[col] = np.log(df[col] / df[col].shift(1)) * 100 #log transform
    df = df.copy().loc[(df.index <= cutoff) & (df.index >= start)] #within␣
 ↪relevant period
    df.dropna(axis=0, inplace=True)
    return df

BTC = log_returns(BTC, 'BTC-USD')
ETH = log_returns(ETH, 'ETH-USD')
```

4 c)

[13]:
```
def run_lin_reg(df, col, constant=True):
    #maybe move into helper function
    df['y_actual'] = df[col] - df['FFR']
    df['x'] = df['SP500'] - df['FFR']
    y = df['y_actual'] # potentially skip df[y_actual] step ?
    X = df['x']
    if constant == True:
        X = sm.add_constant(X)
        model = sm.OLS(y, X).fit()
        df['y_hat'] = model.predict(X)
```

```
        df['residual'] = df['y_hat'] - df['y_actual']
    elif constant == False:
        model = sm.OLS(y, X).fit()
        df['y_hat_no_const'] = model.predict(X)
        df['residual_no_const'] = df['y_hat'] - df['y_actual']
    print(f'Results for {col}:\n', model.summary())
    return model



BTC_model = run_lin_reg(BTC, 'BTC-USD')
ETH_model = run_lin_reg(ETH, 'ETH-USD')
```

```
Results for BTC-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.159
Model:                            OLS   Adj. R-squared:                  0.159
Method:                 Least Squares   F-statistic:                     367.2
Date:                Wed, 11 Oct 2023   Prob (F-statistic):           4.34e-75
Time:                        11:29:46   Log-Likelihood:                -5578.2
No. Observations:                1941   AIC:                         1.116e+04
Df Residuals:                    1939   BIC:                         1.117e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1682      0.119      1.412      0.158      -0.065       0.402
x              1.0094      0.053     19.162      0.000       0.906       1.113
==============================================================================
Omnibus:                      277.877   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3384.188
Skew:                          -0.195   Prob(JB):                         0.00
Kurtosis:                       9.457   Cond. No.                         2.97
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Results for ETH-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.174
Model:                            OLS   Adj. R-squared:                  0.174
Method:                 Least Squares   F-statistic:                     296.0
```

```
Date:                Wed, 11 Oct 2023   Prob (F-statistic):           2.48e-60
Time:                        11:29:46   Log-Likelihood:                -4373.9
No. Observations:                1405   AIC:                             8752.
Df Residuals:                    1403   BIC:                             8762.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3674      0.184      1.991      0.047       0.006       0.729
x              1.2264      0.071     17.203      0.000       1.087       1.366
==============================================================================
Omnibus:                      245.075   Durbin-Watson:                   2.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2226.082
Skew:                          -0.526   Prob(JB):                         0.00
Kurtosis:                       9.076   Cond. No.                         3.49
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

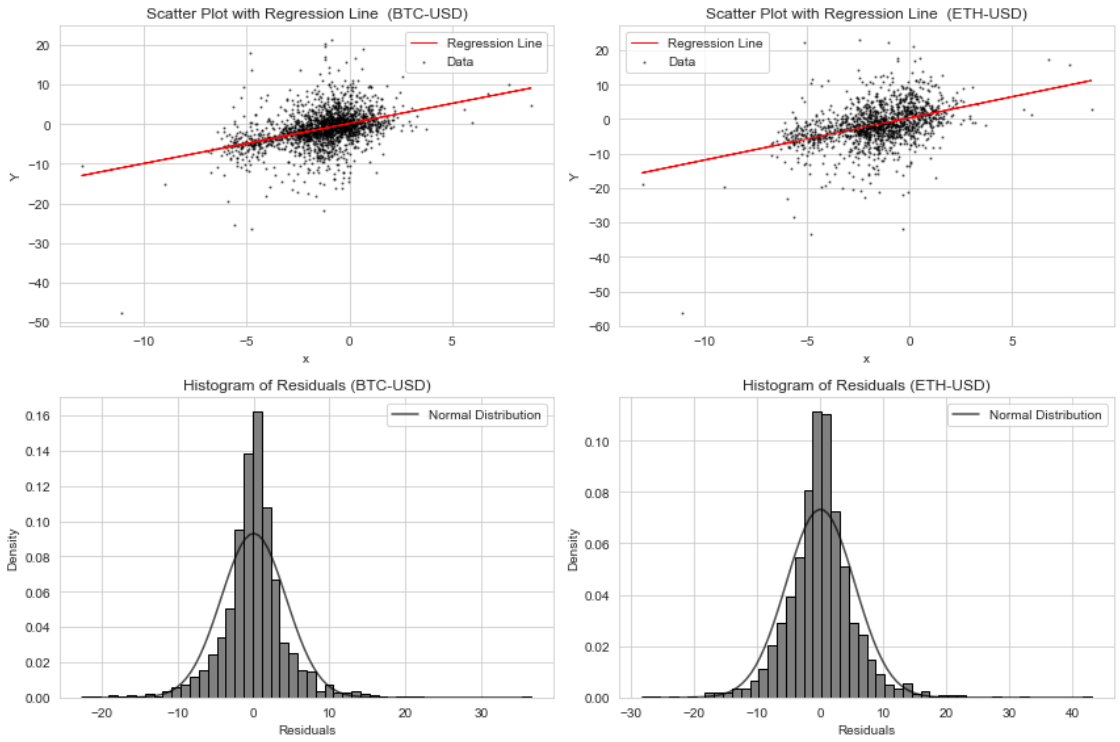
[132]:
```
mpl.rcParams['pdf.fonttype'] = 42
mpl.rcParams['ps.fonttype'] = 42
mpl.rcParams['font.family'] = 'Arial'
```

[363]:
```
def plot_reg_resid(crypto1, crypto2, robust = False, constant = True,␣
 ↪save=True):

    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

    for i, df in enumerate([crypto1, crypto2]):
        if robust == True:
            rbst = 'Robust ' #for titles
            const = ''
            y_hat = 'y_hat_robust'
            residual = 'residual_robust'
        elif constant == False:
            rbst = ''
            const = '(without constant)'
            y_hat = 'y_hat_no_const'
            residual = 'residual_no_const'
        else:
            rbst = ''
            const = ''
            y_hat = 'y_hat'
            residual = 'residual'
```

```
        row = 0

        col = i % 2
        # scatter plot
        axes[row, col].scatter(df['x'], df['y_actual'], c = 'black',marker='.',␣
↪s=5,alpha = 0.5, label = 'Data')
        # regression line
        axes[row, col].plot(df['x'], df[y_hat], color='red', linewidth=1,␣
↪label='Regression Line')

        # titles and labels
        axes[row, col].set_title(f'Scatter Plot with {rbst}Regression Line␣
↪{const} ({df.columns[0]})')
        axes[row, col].set_xlabel('x')
        axes[row, col].set_ylabel('Y')
        axes[row, col].legend()

    # histograms of residuals
        row = 1  # Second row
        col = i   # Calculate column position

        axes[row, col].hist(df[residual], bins=50, color='gray',␣
↪edgecolor='black', alpha=1, density=True, zorder=1)  # Normalize histogram

        # overlay normal dist. curve
        mu, std = df[residual].mean(), df[residual].std()
        x = np.linspace(df[residual].min(), df[residual].max(), 100)
        pdf = stats.norm.pdf(x, mu, std)
        axes[row, col].plot(x, pdf, color='black', alpha =0.7, label='Normal␣
↪Distribution', zorder=2)

        axes[row, col].set_title(f'Histogram of Residuals ({df.columns[0]})')
        axes[row, col].set_xlabel('Residuals')
        axes[row, col].set_ylabel('Density')
        axes[row, col].legend()

    plt.tight_layout()
    if save == True:
        if constant==True:
            plt.savefig(r"/Users/tomonuallain/Desktop/Final_Year/FIN30200/
↪reg_resid_nonrobust.jpeg", dpi=300, bbox_inches='tight')
        else:
            plt.savefig(r"/Users/tomonuallain/Desktop/Final_Year/FIN30200/
↪reg_resid_nonrobust_noConstant.jpeg", dpi=300, bbox_inches='tight')
```

```
    plt.show()

# plt.savefig(r"/Users/tomonuallain/Desktop/Final_Year/FIN30200")
```

[365]:
```
plot_reg_resid(BTC, ETH, save=True)
```



[16]:
```
def anderson_darling_test(crypto_name, residuals, significance_level=0.05):

    result = stats.anderson(residuals, dist='norm')

    # Extract test statistic, critical values, and significance levels
    test_statistic = result.statistic
    critical_values = result.critical_values
    significance_levels = result.significance_level

    # Find the critical value corresponding to your chosen significance level
    critical_value = None
    for i, alpha in enumerate(significance_levels):
        if alpha == significance_level*100:
            critical_value = critical_values[i]
            break
```

```
    # Check if the test statistic is less than the critical value at the chosen␣
↪significance level
    exceeds_critical = test_statistic > critical_value
    text_1 = "exceeds" if exceeds_critical else "does not exceed"
    text_2 = "" if exceeds_critical else "not"
    print(f"Anderson-Darling Test for Normality ({crypto_name})")
    print("----------------------------------------")
    print(f"Test Statistic: {test_statistic:.4f}")
    print(f"Critical Value ({significance_level}): {critical_value:.4f}")
    print(f"The value of the statistic {text_1} the critical value, so for␣
↪{crypto_name},\nthe null hypothesis may{text_2} be rejected (at␣
↪{significance_level} significance level)")
```

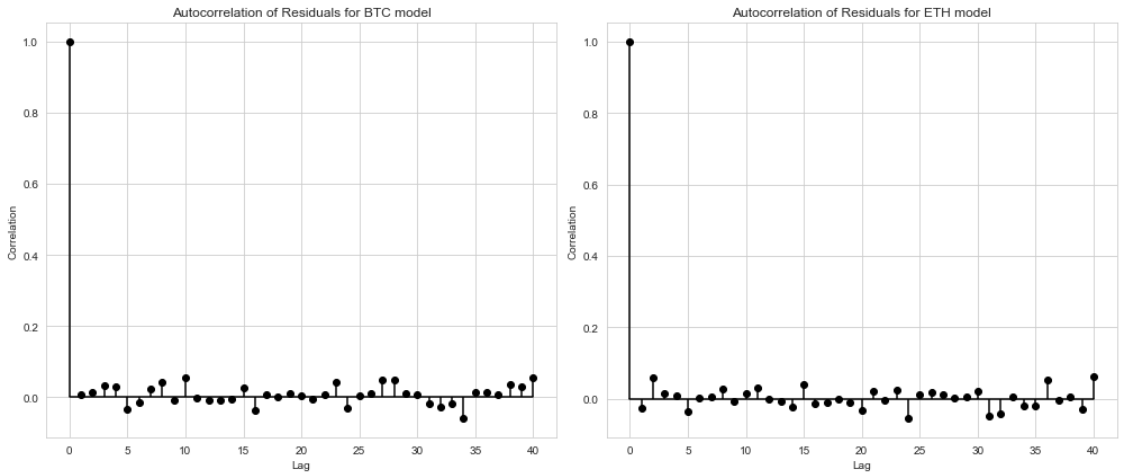
[17]:
```
anderson_darling_test('BTC', BTC['residual'])
anderson_darling_test('ETH', ETH['residual'])
```

```
Anderson-Darling Test for Normality (BTC)
----------------------------------------
Test Statistic: 38.6812
Critical Value (0.05): 0.7850
The value of the statistic exceeds the critical value, so for BTC,
the null hypothesis may be rejected (at 0.05 significance level)
Anderson-Darling Test for Normality (ETH)
----------------------------------------
Test Statistic: 17.9303
Critical Value (0.05): 0.7850
The value of the statistic exceeds the critical value, so for ETH,
the null hypothesis may be rejected (at 0.05 significance level)
```

Test for serial correlation

[366]:
```
def acf_chart(crypto_name1, residuals1, crypto_name2, residuals2):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

    # calculate the autocorrelation function
    acf_result1 = sm.tsa.acf(residuals1, fft=True)
    ax1.stem(range(len(acf_result1)), acf_result1, basefmt='black',␣
↪linefmt='black', markerfmt='ko')

    ax1.set_title(f"Autocorrelation of Residuals for {crypto_name1} model")
    ax1.set_xlabel("Lag")
    ax1.set_ylabel("Correlation")

    acf_result2 = sm.tsa.acf(residuals2, fft=True)
    ax2.stem(range(len(acf_result2)), acf_result2, basefmt='black',␣
↪linefmt='black', markerfmt='ko')

    ax2.set_title(f"Autocorrelation of Residuals for {crypto_name2} model")
```

```
    ax2.set_xlabel("Lag")
    ax2.set_ylabel("Correlation")


    plt.tight_layout()
    plt.show()

acf_chart("BTC", BTC['residual'], "ETH", ETH['residual'])
```



examination of the above charts would not lead one to believe there is much autocorrelation

[22]:
```
def breusch_godfrey_test(crypto_name, residuals, lags=None):

    # Breusch-Godfrey test
    test_results = acorr_breusch_godfrey(residuals, nlags=lags)

    LM_statistic = test_results[0]
    p_value = test_results[1]
    print(f"The {crypto_name} model has an LM statistic of {round(LM_statistic,␣
 ↪2)}, which corresponds to a p-value of {round(p_value, 2)}")
    return LM_statistic, p_value

breusch_godfrey_test('BTC', BTC_model)
breusch_godfrey_test('ETH', ETH_model)
```

```
The BTC model has an LM statistic of 26.64, which corresponds to a p-value of
0.37
The ETH model has an LM statistic of 16.59, which corresponds to a p-value of
0.83
```

[22]:
```
(16.59283879908361, 0.8287414663719603)
```

[23]:
```
def ljung_box_test(ts_data, lags=20):
    # perform the test
    lb_test_results = acorr_ljungbox(ts_data, lags=lags)

    test_statistics = lb_test_results[0]
    p_values = lb_test_results[1]

    # table with the results
    results_table = []
    for lag, test_statistic, p_value in zip(range(1, lags + 1),␣
 ↪test_statistics, p_values):
        results_table.append([lag, test_statistic, p_value])

    # print table
    headers = ["Lag", "Test Statistic", "p-value"]
    print(tabulate(results_table, headers, tablefmt="fancy_grid"))

    return test_statistics, p_values

ljung_box_test(BTC['residual'])
```

Lag	Test Statistic	p-value
1	0.076675	0.781855
2	0.470143	0.790514
3	2.42894	0.48827
4	3.99538	0.406631
5	6.12175	0.294552
6	6.61074	0.358349
7	7.59896	0.369281
8	11.2574	0.187539
9	11.3707	0.25115
10	17.0411	0.0734589
11	17.0502	0.1064
12	17.187	0.142698

```
13          17.3138     0.185354

14          17.3688     0.237057

15          18.7545     0.225076

16          21.3707     0.164697

17          21.5017     0.20465

18          21.5017     0.254859

19          21.7046     0.299204

20          21.7282     0.355441
```

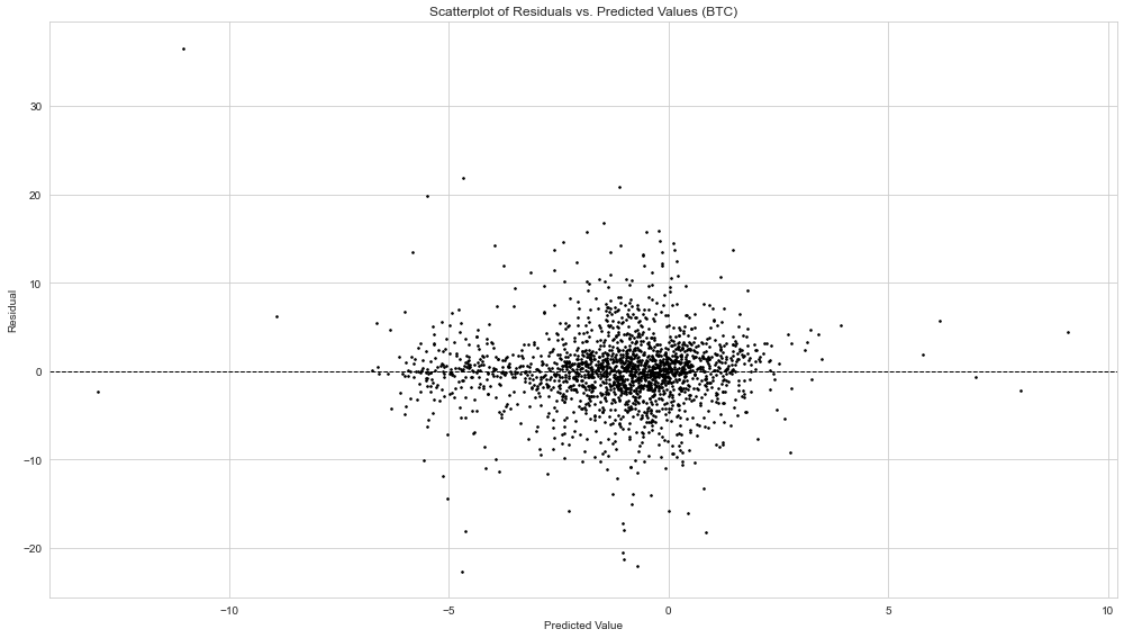
[23]:
```
(array([ 0.07667498,  0.47014348,  2.42893915,  3.99538419,  6.12174521,
         6.61074358,  7.59895537, 11.25736347, 11.37073263, 17.04112529,
        17.05018435, 17.18699101, 17.31378768, 17.36879297, 18.75447873,
        21.37070716, 21.5016835 , 21.50174123, 21.70459307, 21.72818176]),
 array([0.78185499, 0.79051414, 0.48827034, 0.40663089, 0.29455222,
        0.3583488 , 0.36928108, 0.18753869, 0.25115035, 0.07345892,
        0.10639988, 0.14269769, 0.1853537 , 0.23705699, 0.22507609,
        0.16469713, 0.20465   , 0.25485895, 0.29920393, 0.35544053]))
```

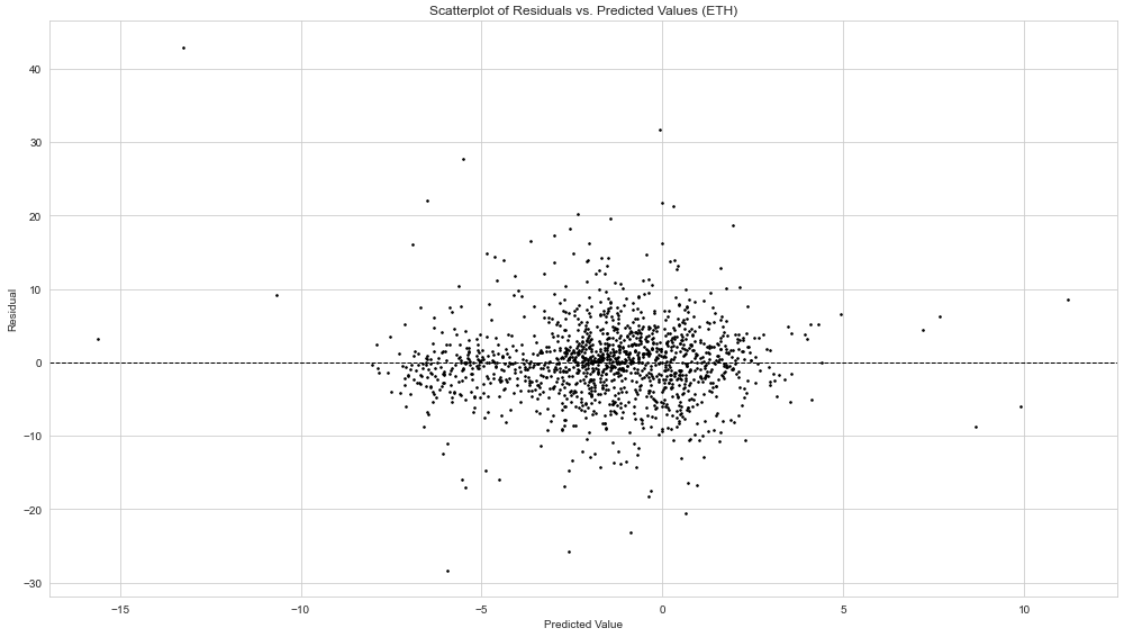
Testing for Heteroskedasticity

Plot of residuals vs fitted values to test for hetersoskedasticity - verify the assumption that the residuals are randomly distributed and have constant variance

[367]:
```python
def plot_residuals_fitted_vals(crypto_name, predicted, residuals):

    # scatterplot
    plt.figure(figsize=(14, 8))
    plt.scatter(predicted, residuals, marker='.', color='black', alpha=1, s=10)

    # labels and a title
    plt.xlabel("Predicted Value")
    plt.ylabel("Residual")
    plt.title(f"Scatterplot of Residuals vs. Predicted Values ({crypto_name})")

    # horizontal line at y=0 for reference
    plt.axhline(y=0, color='black', linestyle='--', linewidth=1)

    plt.tight_layout()
```

```
    plt.savefig(f"//Users//tomonuallain//Desktop//Final_Year//FIN30200//
↪resid_pred_{crypto_name}.jpeg", dpi=300, bbox_inches='tight')
    plt.show()

plot_residuals_fitted_vals('BTC', BTC['y_hat'], BTC['residual'])
plot_residuals_fitted_vals('ETH', ETH['y_hat'], ETH['residual'])
```





White test

[25]:
```
def white_test(crypto_name, crypto_model, df, residuals):

    # perform the White test
    exog = crypto_model.model.exog

    white_test_result = het_white(residuals, exog)

    test_statistic = white_test_result[0]
    p_value = white_test_result[1]

    print(f"White Test for Heteroskedasticity ({crypto_name})")
    print("---------------------------------------")
    print(f"Test Statistic: {test_statistic:.4f}")
    print(f"P-Value: {p_value:.4f}\n\n")

    return test_statistic, p_value

white_test('BTC', BTC_model, BTC, BTC['residual'])
white_test('ETH', ETH_model, ETH, ETH['residual'])
```

```
White Test for Heteroskedasticity (BTC)
---------------------------------------
Test Statistic: 51.9855
P-Value: 0.0000


White Test for Heteroskedasticity (ETH)
---------------------------------------
Test Statistic: 41.2578
P-Value: 0.0000
```

[25]:
```
(41.25780575391694, 1.0989585515201661e-09)
```

[26]:
```
def breusch_pagan_test(crypto_name, crypto_model, df, residuals):
    exog = crypto_model.model.exog

    # perform the Breusch-Pagan test
    bp_test_result = het_breuschpagan(residuals, exog, robust=False)

    test_statistic = bp_test_result[0]
    p_value = bp_test_result[1]
```

```
    print(f"Breusch-Pagan Test for Heteroskedasticity ({crypto_name})")
    print("-----------------------------------------------")
    print(f"Test Statistic: {test_statistic:.8f}")
    print(f"P-Value: {p_value:.4f}")


breusch_pagan_test('BTC', BTC_model, BTC, BTC['residual'])
breusch_pagan_test('ETH', ETH_model, ETH, ETH['residual'])
```

```
Breusch-Pagan Test for Heteroskedasticity (BTC)
-----------------------------------------------
Test Statistic: 65.72110376
P-Value: 0.0000
Breusch-Pagan Test for Heteroskedasticity (ETH)
-----------------------------------------------
Test Statistic: 29.47135347
P-Value: 0.0000
```

5 d)

[27]:
```
def run_lin_reg_robust_se(df, col, const=True):

    y = df['y_actual'] # potentially skip df[y_actual] step ?
    X = df['x']
    if const == True:
        X = sm.add_constant(X)
    new_model = sm.OLS(y, X).fit(cov_type='HAC', cov_kwds={'maxlags':12,␣
↪'use_correction':True})
    df['y_hat_robust'] = new_model.predict(X)
    df['residual_robust'] = df['y_hat_robust'] - df['y_actual']
    print(f'\n\nResults for {col}:\n\n', new_model.summary())
    return new_model

BTC_model_robust = run_lin_reg_robust_se(BTC, 'BTC-USD')
ETH_model_robust = run_lin_reg_robust_se(ETH, 'ETH-USD')
```

```
Results for BTC-USD:

                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.159
Model:                            OLS   Adj. R-squared:                  0.159
Method:                 Least Squares   F-statistic:                     222.1
Date:                Wed, 11 Oct 2023   Prob (F-statistic):           1.25e-47
Time:                        11:29:55   Log-Likelihood:                -5578.2
```

No. Observations:	1941	AIC:	1.116e+04
Df Residuals:	1939	BIC:	1.117e+04
Df Model:	1		
Covariance Type:	HAC		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.1682	0.124	1.354	0.176	-0.075	0.412
x	1.0094	0.068	14.902	0.000	0.877	1.142

Omnibus:	277.877	Durbin-Watson:	1.987
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3384.188
Skew:	-0.195	Prob(JB):	0.00
Kurtosis:	9.457	Cond. No.	2.97

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 12 lags and with small sample correction

Results for ETH-USD:

OLS Regression Results

Dep. Variable:	y_actual	R-squared:	0.174
Model:	OLS	Adj. R-squared:	0.174
Method:	Least Squares	F-statistic:	160.8
Date:	Wed, 11 Oct 2023	Prob (F-statistic):	5.89e-35
Time:	11:29:55	Log-Likelihood:	-4373.9
No. Observations:	1405	AIC:	8752.
Df Residuals:	1403	BIC:	8762.
Df Model:	1		
Covariance Type:	HAC		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3674	0.186	1.971	0.049	0.002	0.733
x	1.2264	0.097	12.679	0.000	1.037	1.416

Omnibus:	245.075	Durbin-Watson:	2.052
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2226.082
Skew:	-0.526	Prob(JB):	0.00
Kurtosis:	9.076	Cond. No.	3.49

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC)

using 12 lags and with small sample correction

[28]:
```
ETH_model_robust.pvalues[0].round(8)
```

[28]:
```
0.04874137
```

6 e)

[29]:
```
# testing beta=1
hypotheses = 'x = 1'

for model in ([BTC_model_robust, 'BTC'], [ETH_model_robust, 'ETH']):
    t_stat = model[0].t_test(hypotheses)
    print(model[1])
    print(t_stat,'\n\n')
```

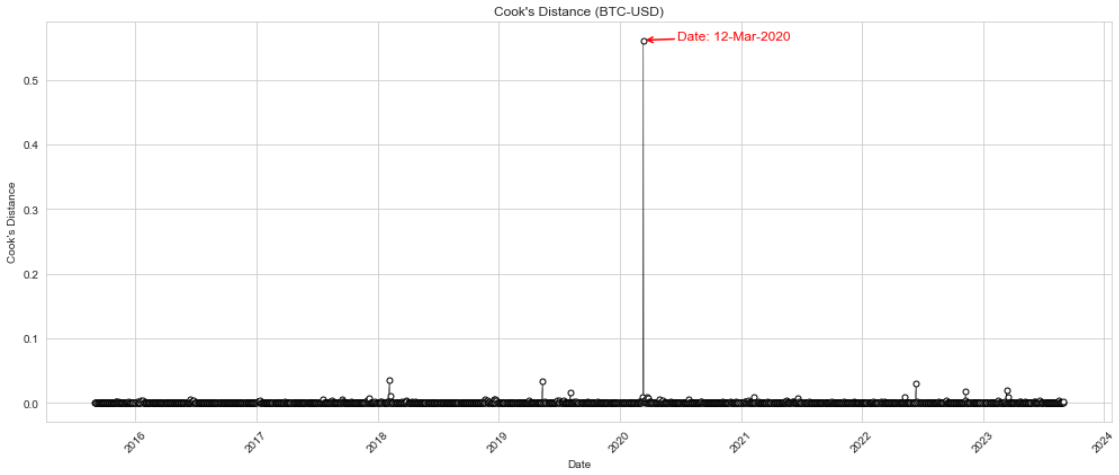
```
BTC
                             Test for Constraints
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
c0             1.0094      0.068      0.139      0.890       0.877       1.142
==============================================================================


ETH
                             Test for Constraints
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
c0             1.2264      0.097      2.341      0.019       1.037       1.416
==============================================================================
```

[465]:
```
print(BTC['BTC-USD'].std())
print(ETH['ETH-USD'].std())
print(BTC['SP500'].std())
```

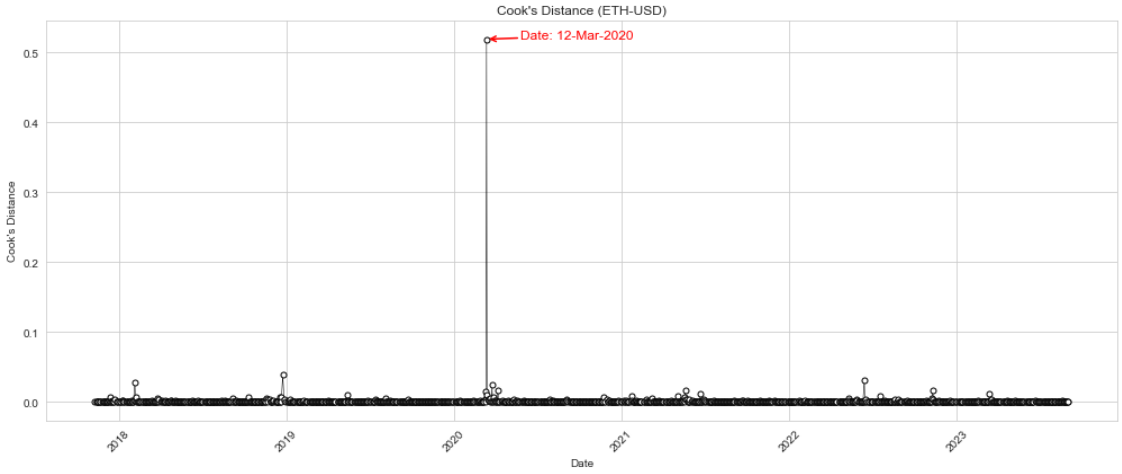
```
4.412909240291066
5.738605895215906
1.1901508448348819
```

7 f)

[155]:
```
def model_influence(model, crypto_df):
    # Calculate Cook's distance
    infl = model.get_influence()
    cook_d = infl.cooks_distance[0]  # Cook's distance is the first element in␣
↪the tuple
    cook_df = pd.DataFrame({'Date': crypto_df.index, "CooksDistance": cook_d})
    idx_max = cook_df["CooksDistance"].idxmax()
    max_cook = cook_df["CooksDistance"].max()
    date_max = crypto_df.index[idx_max]



    # plot
    plt.figure(figsize=(14, 6))
    plt.plot(cook_df['Date'], cook_df["CooksDistance"], markeredgecolor='k',␣
↪marker = 'o', markersize=5,
             markerfacecolor='white', linestyle='-', color='black',
             linewidth=0.5)

    plt.annotate(f"Date: {date_max.strftime('%d-%b-%Y')}", (date_max, max_cook),
             xytext=(30, 0), textcoords='offset points',
             arrowprops=dict(arrowstyle='->', lw=1.5, color='red', shrinkA=0),
             fontsize=12, color='red')

    plt.title(f"Cook's Distance ({crypto_df.columns[0]})")
    plt.xlabel('Date')
    plt.ylabel("Cook's Distance")
    plt.xticks(rotation=45)
    plt.tight_layout()

    plt.savefig(f"cooksDistance{crypto_df.columns[0]}.jpeg", dpi=300,␣
↪bbox_inches='tight')
    plt.show()
    print(f"Observation with highest model influence for {crypto_df.columns[0]}:
↪ {date_max}")

model_influence(BTC_model, BTC)
model_influence(ETH_model, ETH)
```




```
Observation with highest model influence for BTC-USD: 2020-03-12 00:00:00
```



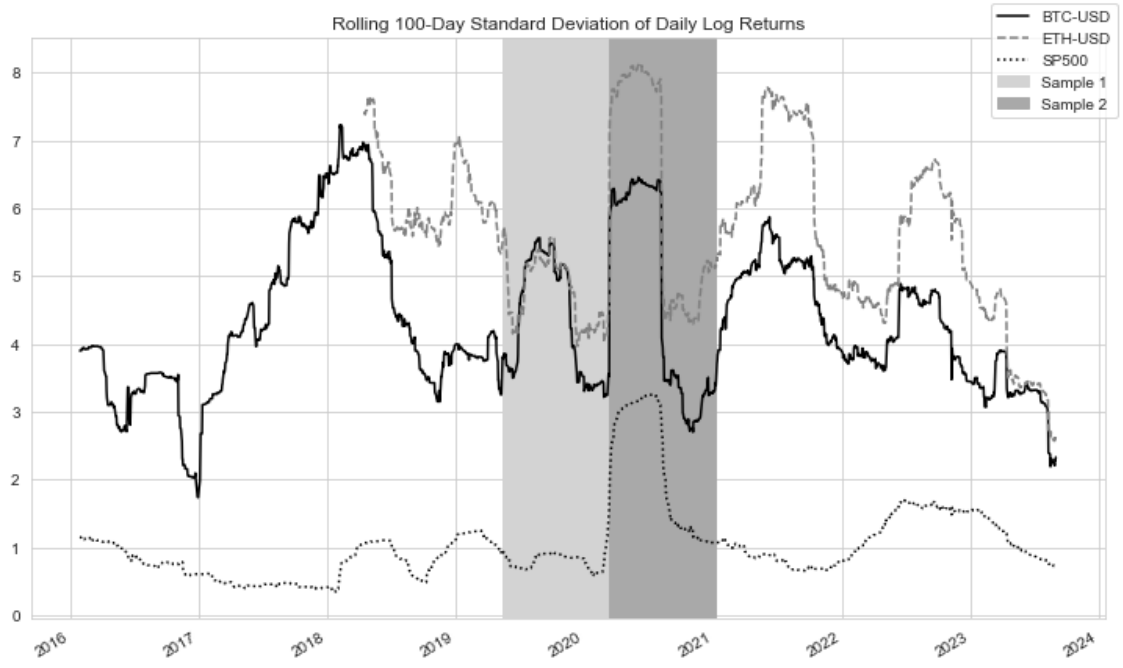

```
Observation with highest model influence for ETH-USD: 2020-03-12 00:00:00
```

[152]:
```
timestamp = BTC.index[1097]
timestamp.strftime('%d-%b-%Y')
# datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')
```

[152]:
```
'12-Mar-2020'
```

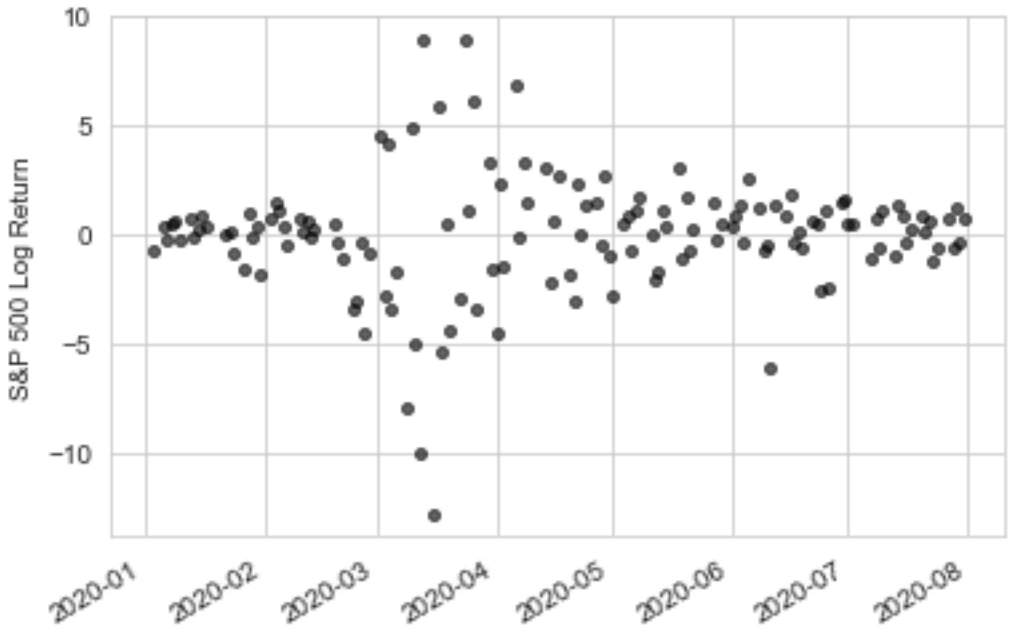
[37]:
```
def plot_volatility(df1, df2, col1, col2, col3, n=100):
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.set_title('Rolling 100-Day Standard Deviation of Daily Log Returns')

    df1[col1].rolling(n).std().plot(label=col1, alpha=1, c='black', ax=ax)
```

```
    df2[col2].rolling(n).std().plot(label=col2, alpha=1, c='gray', style='--',␣
 ↪ax=ax)
    df1[col3].rolling(n).std().plot(label=col3, alpha=1, c='black', style=':',␣
 ↪ax=ax)
    # Shade the first time period in light grey
    ax.axvspan('2019-05-15', '2020-03-11', alpha=1, color='lightgrey',␣
 ↪label='Sample 1')

    # Shade the second time period in dark grey
    ax.axvspan('2020-03-12', '2021-01-05', alpha=1, color='darkgrey',␣
 ↪label='Sample 2')
#     fig.tight_layout()
    fig.legend()
    fig.tight_layout()

    plt.savefig('std_dev_chart.jpeg', dpi=300, bbox_inches='tight')

    plt.show()


plot_volatility(BTC, ETH, 'BTC-USD', 'ETH-USD', 'SP500')
```



[467]:
```
BTC.loc['2020-01-01':'2020-08-01', 'SP500'].plot(style='o', c = 'black',␣
 ↪markersize=4, alpha=0.6)
plt.ylabel('S&P 500 Log Return')
```

```
plt.show()
```



[258]:
```
BTC.loc['2020-03-01':'2020-03-31', 'SP500']
```

[258]:
```
2020-03-02     4.501087
2020-03-03    -2.851053
2020-03-04     4.133635
2020-03-05    -3.451073
2020-03-06    -1.720100
2020-03-09    -7.901039
2020-03-10     4.821508
2020-03-11    -5.010286
2020-03-12    -9.994485
2020-03-13     8.880836
2020-03-16   -12.765214
2020-03-17     5.822629
2020-03-18    -5.322234
2020-03-19     0.469685
2020-03-20    -4.432763
2020-03-23    -2.973149
2020-03-24     8.968316
2020-03-25     1.146900
2020-03-26     6.054383
2020-03-27    -3.426781
2020-03-30     3.296662
2020-03-31    -1.614238
```

```
Name: SP500, dtype: float64
```

[41]:
```
print(BTC['SP500'].idxmin(), BTC['SP500'].min())
```

```
2020-03-16 00:00:00 -12.765214115647328
```

[42]:
```
print(BTC['BTC-USD'].idxmin(), BTC['BTC-USD'].min())
```

```
2020-03-12 00:00:00 -46.4730175339773
```

[43]:
```
print(ETH['ETH-USD'].idxmin(), ETH['ETH-USD'].min())
```

```
2020-03-12 00:00:00 -55.073174382201074
```

[240]:
```
def get_subSamples_and_regress(df, col, date='2020-03-12', n=200):
    df_1 = df[df.index < date].tail(200).copy() #gets the n observations before␣
 ↪the given date
    df_2 = df[df.index >= date].head(200).copy() #gets the n observations after␣
 ↪and incl. the given date
    df_joined = df_1.append(df_2)
    df_joined.sort_index(inplace=True)

    model_1 = run_lin_reg(df_1, col)
    model_2 = run_lin_reg(df_2, col)
    model_joined = run_lin_reg(df_joined, col)

    return df_1, df_2, model_1, model_2, model_joined
```

BTC subsample regressions:

[241]:
```
BTC_1, BTC_2, BTC_model1, BTC_model2, BTC_modelJoined=␣
 ↪get_subSamples_and_regress(BTC, 'BTC-USD')
```

```
Results for BTC-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                    0.7418
Date:                Thu, 12 Oct 2023   Prob (F-statistic):              0.390
Time:                        13:01:16   Log-Likelihood:                -577.16
No. Observations:                 200   AIC:                             1158.
Df Residuals:                     198   BIC:                             1165.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
const         -1.5399      0.552     -2.792      0.006      -2.628      -0.452
x              0.2062      0.239      0.861      0.390      -0.266       0.678
==============================================================================
Omnibus:                       18.843   Durbin-Watson:                   1.903
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               57.564
Skew:                          -0.256   Prob(JB):                     3.16e-13
Kurtosis:                       5.578   Cond. No.                         4.69
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Results for BTC-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.235
Method:                 Least Squares   F-statistic:                     62.21
Date:                Thu, 12 Oct 2023   Prob (F-statistic):           2.04e-13
Time:                        13:01:16   Log-Likelihood:                -579.46
No. Observations:                 200   AIC:                             1163.
Df Residuals:                     198   BIC:                             1170.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.5475      0.312      1.756      0.081      -0.067       1.162
x              1.1195      0.142      7.887      0.000       0.840       1.399
==============================================================================
Omnibus:                      144.319   Durbin-Watson:                   1.717
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4161.864
Skew:                          -2.240   Prob(JB):                         0.00
Kurtosis:                      24.894   Cond. No.                         2.20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Results for BTC-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_actual   R-squared:                       0.171
Model:                            OLS   Adj. R-squared:                  0.169
Method:                 Least Squares   F-statistic:                     82.01
Date:                Thu, 12 Oct 2023   Prob (F-statistic):           6.15e-18
Time:                        13:01:16   Log-Likelihood:                -1163.2
No. Observations:                 400   AIC:                             2330.
```

```
Df Residuals:                      398   BIC:                             2338.
Df Model:                            1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2478      0.244      1.017      0.310      -0.231       0.727
x              0.9801      0.108      9.056      0.000       0.767       1.193
==============================================================================
Omnibus:                      165.236   Durbin-Watson:                   2.040
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2988.025
Skew:                          -1.267   Prob(JB):                         0.00
Kurtosis:                      16.148   Cond. No.                         2.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

ETH subsample regressions:

[242]:
```
ETH_1, ETH_2, ETH_model1, ETH_model2, ETH_modelJoined =␣
↪get_subSamples_and_regress(ETH, 'ETH-USD')
```

```
Results for ETH-USD:
                            OLS Regression Results
==============================================================================
Dep. Variable:                y_actual   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.016
Method:                 Least Squares   F-statistic:                     4.203
Date:                Thu, 12 Oct 2023   Prob (F-statistic):             0.0417
Time:                        13:01:17   Log-Likelihood:                -601.39
No. Observations:                 200   AIC:                             1207.
Df Residuals:                     198   BIC:                             1213.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9255      0.623     -1.487      0.139      -2.153       0.302
x              0.5539      0.270      2.050      0.042       0.021       1.087
==============================================================================
Omnibus:                       20.332   Durbin-Watson:                   1.844
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                39.026
Skew:                          -0.505   Prob(JB):                     3.35e-09
Kurtosis:                       4.914   Cond. No.                         4.69
==============================================================================
```

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Results for ETH-USD:

OLS Regression Results

Dep. Variable:	y_actual	R-squared:	0.260
Model:	OLS	Adj. R-squared:	0.256
Method:	Least Squares	F-statistic:	69.47
Date:	Thu, 12 Oct 2023	Prob (F-statistic):	1.29e-14
Time:	13:01:17	Log-Likelihood:	-628.54
No. Observations:	200	AIC:	1261.
Df Residuals:	198	BIC:	1268.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5611	0.399	1.408	0.161	-0.225	1.347
x	1.5121	0.181	8.335	0.000	1.154	1.870

Omnibus:	96.525	Durbin-Watson:	1.919
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1347.931
Skew:	-1.425	Prob(JB):	2.00e-293
Kurtosis:	15.395	Cond. No.	2.20

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Results for ETH-USD:

OLS Regression Results

Dep. Variable:	y_actual	R-squared:	0.197
Model:	OLS	Adj. R-squared:	0.195
Method:	Least Squares	F-statistic:	97.82
Date:	Thu, 12 Oct 2023	Prob (F-statistic):	9.07e-21
Time:	13:01:17	Log-Likelihood:	-1235.8
No. Observations:	400	AIC:	2476.
Df Residuals:	398	BIC:	2484.
Df Model:	1		
Covariance Type:	nonrobust		

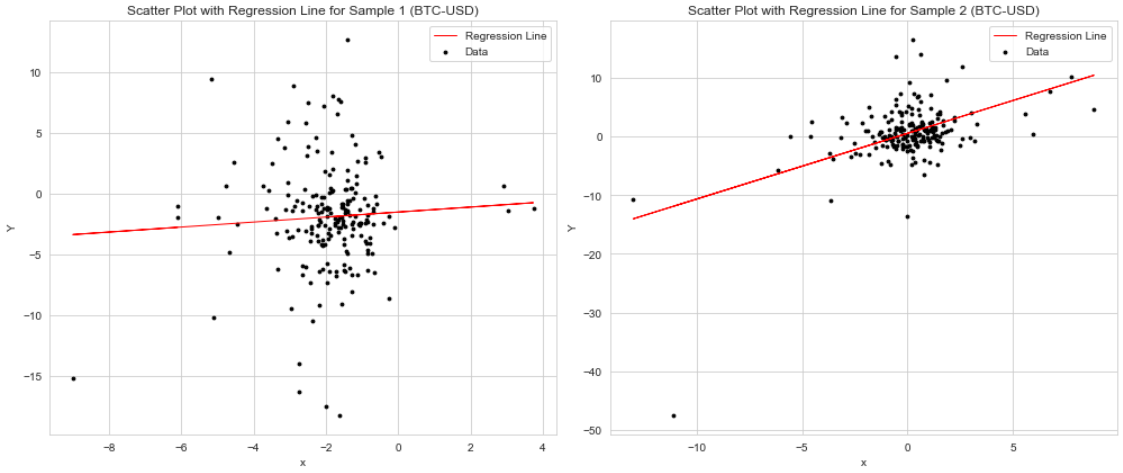
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5223	0.292	1.788	0.075	-0.052	1.097
x	1.2834	0.130	9.890	0.000	1.028	1.539

```
Omnibus:                      146.146   Durbin-Watson:                   2.082
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1860.833
Skew:                          -1.169   Prob(JB):                         0.00
Kurtosis:                      13.304   Cond. No.                         2.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

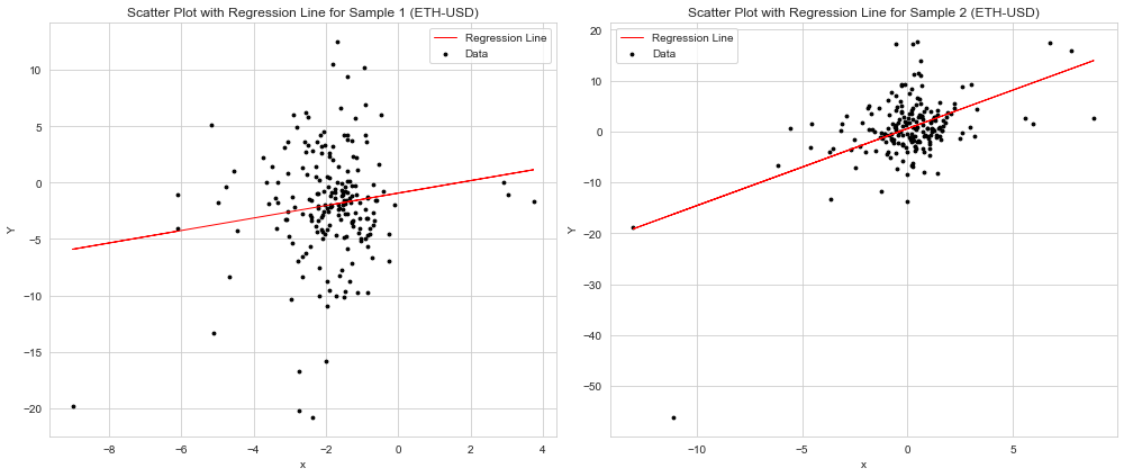
[421]:
```
def plot_sub_sample_reg_resid(df_samp1, df_samp2, save=False, constant=True):

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14,6))

    for i, df in enumerate([df_samp1, df_samp2]):
        row = 0

        col = i % 2
        if constant==True:
            axes[col].scatter(df['x'], df['y_actual'], c = 'black', marker =␣
↪'o', s=8, alpha=1,
                              label = 'Data')
            axes[col].plot(df['x'], df['y_hat'], color='red', linewidth=1,␣
↪label='Regression Line')
            axes[col].set_title(f'Scatter Plot with Regression Line for Sample␣
↪{i+1} ({df.columns[0]})')
        else:
            axes[col].scatter(df['x'], df['y_actual'], c = 'black', marker = '.
↪', s=5, alpha=0.5,
                              label = 'Data')
            axes[col].plot(df['x'], df['y_hat_no_const'], color='red',␣
↪linewidth=1, label='Regression Line')
            axes[col].set_title(f'Scatter Plot with Regression Line for {df.
↪columns[0]} (no constant)')

        axes[col].set_xlabel('x')
        axes[col].set_ylabel('Y')
        axes[col].legend()


    plt.tight_layout()
    if save==True:
        if constant==True:
            plt.savefig(f"subsampRegression{df.columns[0]}", dpi=300,␣
↪bbox_inches='tight')
        else:
            plt.savefig(f"regressionNoConstant", dpi=300, bbox_inches='tight')
```

```
    plt.show()
```

[409]:
```
plot_sub_sample_reg_resid(BTC_1, BTC_2, save=True)
```



[410]:
```
plot_sub_sample_reg_resid(ETH_1, ETH_2, save=True)
```



[250]:
```
def chow_test(model_1, model_2, model_joined, crypto_name):

    # calculate sum of squared residuals for each subsample
    rss_1 = model_1.ssr
    rss_2 = model_2.ssr
    # calculate the sum of squared residuals for the joined period
    rss = model_joined.ssr
```

```
    # calculate degrees of freedom
    k = 2  # no. of model parameters
    n = model_joined.nobs #no. observations
    df1 = k
    df2 = n - (2 * k)

    # calculate Chow test statistic
    numerator = (rss - (rss_1 + rss_2))
    denominator = (rss_1 + rss_2)
    f_statistic = (numerator / denominator) * (df2/df1)

    # calculate critical value from F distribution
    alpha = 0.05  # significance level
    critical_value = f.ppf(1 - alpha, df1, df2)
    p_value = 1-stats.f.cdf(f_statistic, df1, df2)

    # compare F-statistic to critical value
    print(f"{crypto_name}:")
    print('p value: ', round(p_value, 3))
    print('f stat: ', round(f_statistic, 3))
    print('critical value: ', round(critical_value,3))


    if f_statistic > critical_value:
        print("\nThere is a significant structural break.")
    else:
        print("No significant structural break.")
```

[251]:
```
chow_test(BTC_model1, BTC_model2, BTC_modelJoined, 'BTC')
```

```
BTC:
p value:  0.001
f stat:  6.647
critical value:  3.019

There is a significant structural break.
```

[254]:
```
chow_test(ETH_model1, ETH_model2, ETH_modelJoined, 'ETH')
```

```
ETH:
p value:  0.018
f stat:  4.075
critical value:  3.019

There is a significant structural break.
```

F-critical:

[252]:
```
f.ppf(1 - 0.05, 1, 198)
```

[252]:
```
3.8888529328918806
```

8 g)

repeat c) without a constant

[269]:
```
BTC_model_no_const = run_lin_reg(BTC, 'BTC-USD', constant=False)
ETH_model_no_const = run_lin_reg(ETH, 'ETH-USD', constant=False)
```

```
Results for BTC-USD:
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                y_actual   R-squared (uncentered):
0.206
Model:                            OLS   Adj. R-squared (uncentered):
0.206
Method:                 Least Squares   F-statistic:
504.2
Date:                Thu, 12 Oct 2023   Prob (F-statistic):
1.92e-99
Time:                        19:24:21   Log-Likelihood:
-5579.2
No. Observations:                1941   AIC:
1.116e+04
Df Residuals:                    1940   BIC:
1.117e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x              0.9665      0.043     22.453      0.000       0.882       1.051
==============================================================================
Omnibus:                      283.336   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3521.636
Skew:                          -0.209   Prob(JB):                         0.00
Kurtosis:                       9.586   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

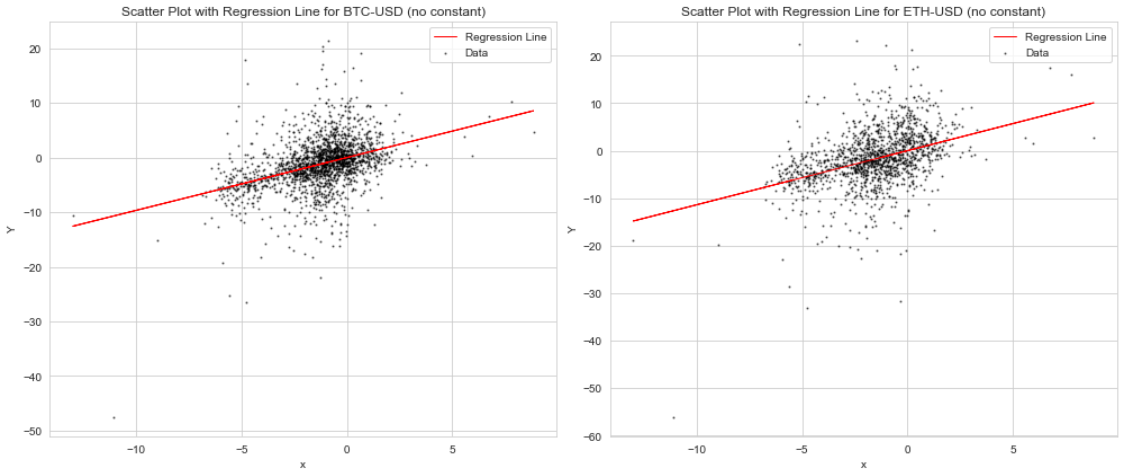
```
Results for ETH-USD:
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                y_actual   R-squared (uncentered):
0.226
Model:                            OLS   Adj. R-squared (uncentered):
0.226
Method:                 Least Squares   F-statistic:
410.7
Date:                Thu, 12 Oct 2023   Prob (F-statistic):
2.67e-80
Time:                        19:24:21   Log-Likelihood:
-4375.9
No. Observations:                1405   AIC:
8754.
Df Residuals:                    1404   BIC:
8759.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x              1.1389      0.056     20.265      0.000       1.029       1.249
==============================================================================
Omnibus:                      253.424   Durbin-Watson:                   2.050
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2379.470
Skew:                          -0.546   Prob(JB):                         0.00
Kurtosis:                       9.281   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

F-critical:

[287]:
```
f.ppf(1-0.05, 1, 1941-1)
```

[287]:
```
3.846256626117546
```

[423]:
```
plot_sub_sample_reg_resid(BTC, ETH, constant=False, save=True)
```



[325]:
```
def log_likelihood_ratio_test(crypto_name, null_model, alt_model, alpha=0.05):

    llfNull = null_model.llf  # log-likelihood of model A
    llfAlt = alt_model.llf  # log-likelihood of model B

    # compute the test stat (log-likelihood ratio)
    log_likelihood_ratio = -2 * (llfNull - llfAlt)

    # chi-squared test with one degree of freedom
    df = 1
    p_value = 1 - chi2.cdf(log_likelihood_ratio, df)
    print(f"P-value for {crypto_name}: {p_value}")
    # Compare the p-value to the significance level
    if p_value < alpha:
        result = "Reject the null hypothesis (H0). The alternative model is a␣
↪better fit."
    else:
        result = "Fail to reject the null hypothesis. The null model is as good␣
↪as the alternative model."

    return result
```

[326]:
```
log_likelihood_ratio_test('BTC', BTC_model, BTC_model_no_const)
```

```
P-value for BTC: 1.0
```

[326]:
```
'Fail to reject the null hypothesis. The null model is as good as the
alternative model.'
```

[328]:
```
log_likelihood_ratio_test('ETH', ETH_model, ETH_model_no_const)
```

```
P-value for ETH: 1.0
```

[328]:
```
'Fail to reject the null hypothesis. The null model is as good as the
alternative model.'
```

AIC vs BIC

[334]:
```
def aic_bic(crypto_name, modelA, modelB):
    print(crypto_name)
    print(f'AIC:\tModel1: {modelA.aic}\tModel2: {modelB.aic}')
    print(f'BIC:\tModel1: {modelA.bic}\tModel2: {modelB.bic}')
```

[473]:
```
aic_bic('\t\t\t\tBTC\n', BTC_model, BTC_model_no_const)
```

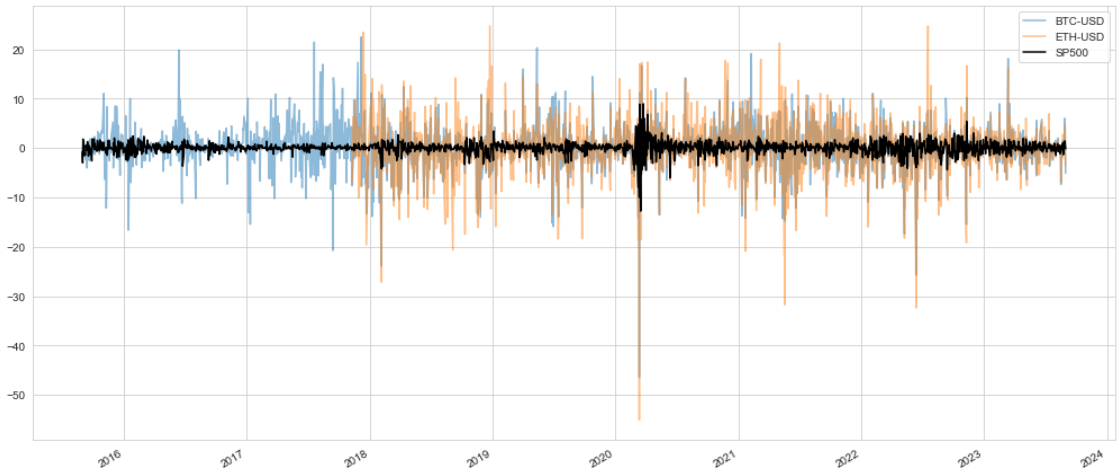
```
                                BTC

AIC:    Model1: 11160.353636930831      Model2: 11160.348631748755
BIC:    Model1: 11171.49555409717       Model2: 11165.919590331923
```

[474]:
```
aic_bic('\t\t\t\tETH\n', ETH_model, ETH_model_no_const)
```

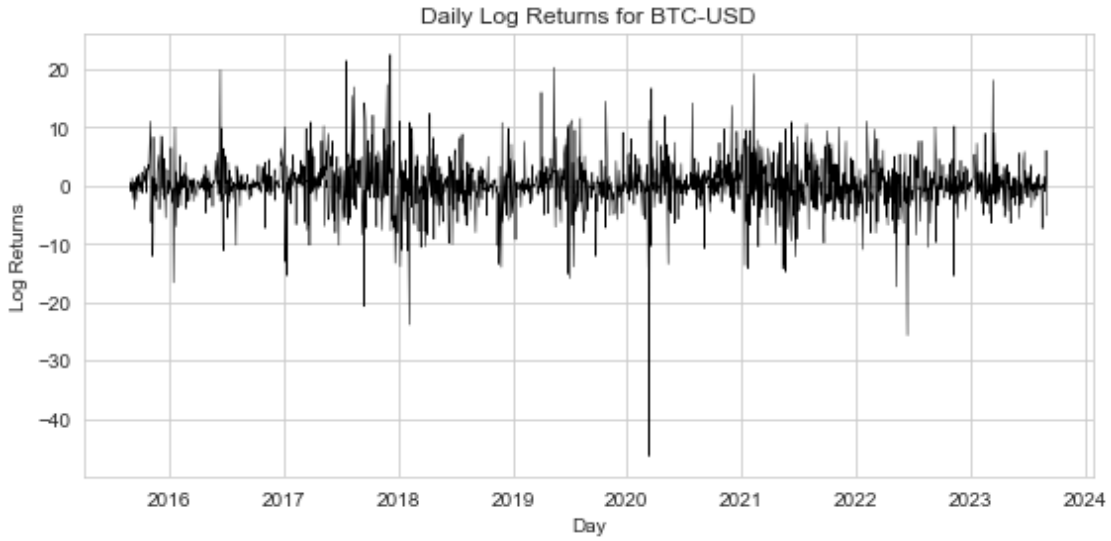
```
                                ETH

AIC:    Model1: 8751.849180576648       Model2: 8753.815262656424
BIC:    Model1: 8762.344765740183       Model2: 8759.063055238192
```

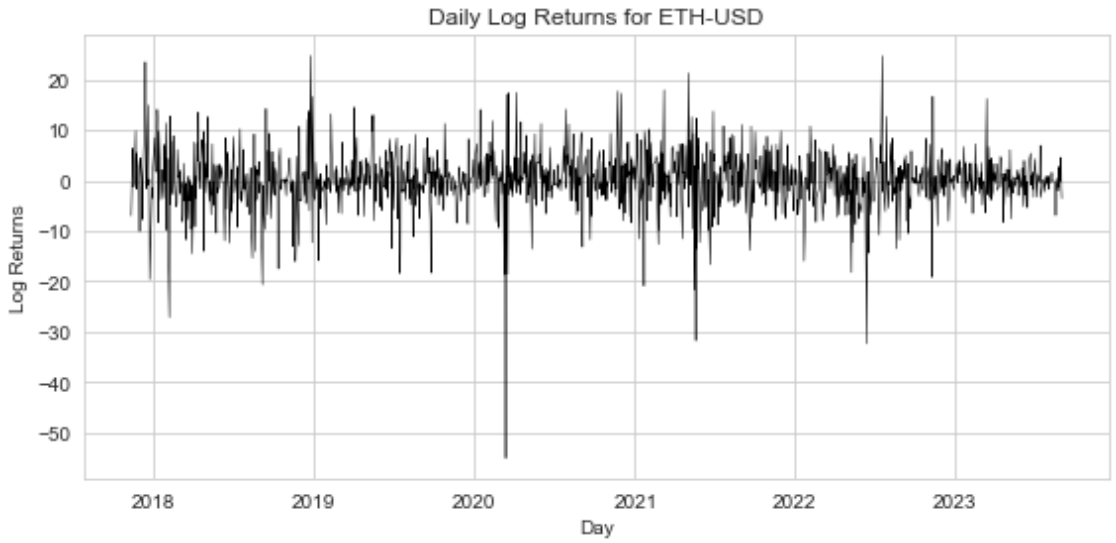
9 Other Code/Misc. Charts

[369]:
```
fig, ax = plt.subplots(figsize=(14, 6))

BTC['BTC-USD'].plot(alpha=0.5, ax=ax)
ETH['ETH-USD'].plot(alpha=0.5, ax=ax)

BTC['SP500'].plot(c='black', ax=ax)
ax.legend()
fig.tight_layout()
```



[372]:
```python
def plot_daily_log_returns(df, column_name):

    fig, ax = plt.subplots(figsize=(8, 4))

    # plot daily log returns as a line
    ax.plot(df[column_name], color='black', linewidth=0.5)

    # add labels and title
    ax.set_xlabel('Day')
    ax.set_ylabel('Log Returns')
    ax.set_title(f'Daily Log Returns for {column_name}')

    # customize tick labels
    ax.tick_params(axis='both', which='both', labelsize=10)

    # Customize the appearance of the plot
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)

    plt.tight_layout()
    plt.show()

plot_daily_log_returns(BTC, 'BTC-USD')
plot_daily_log_returns(ETH, 'ETH-USD')
```

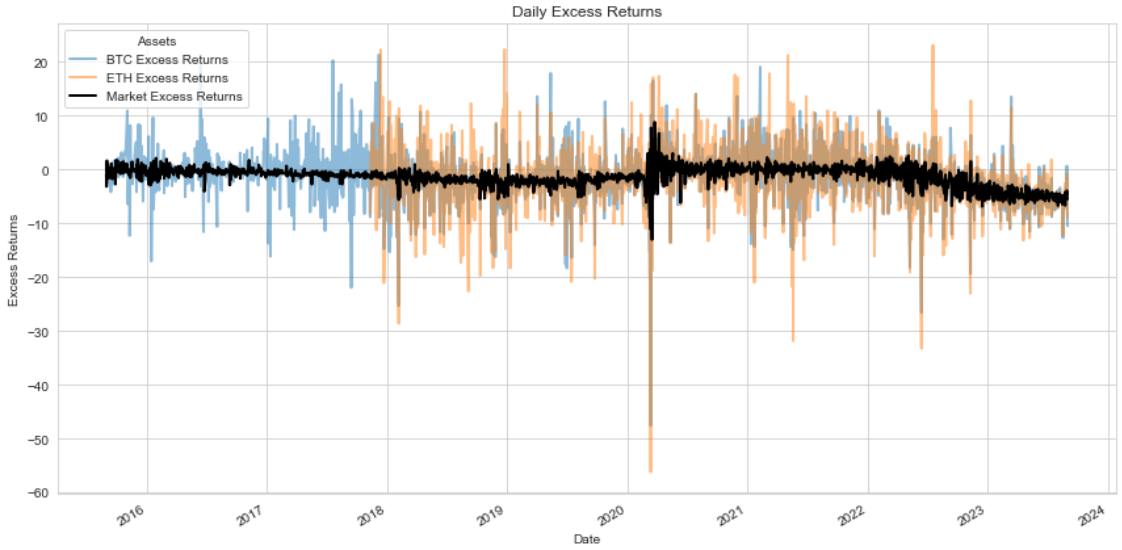




Log Excess Returns for BTC, ETH and S&P 500

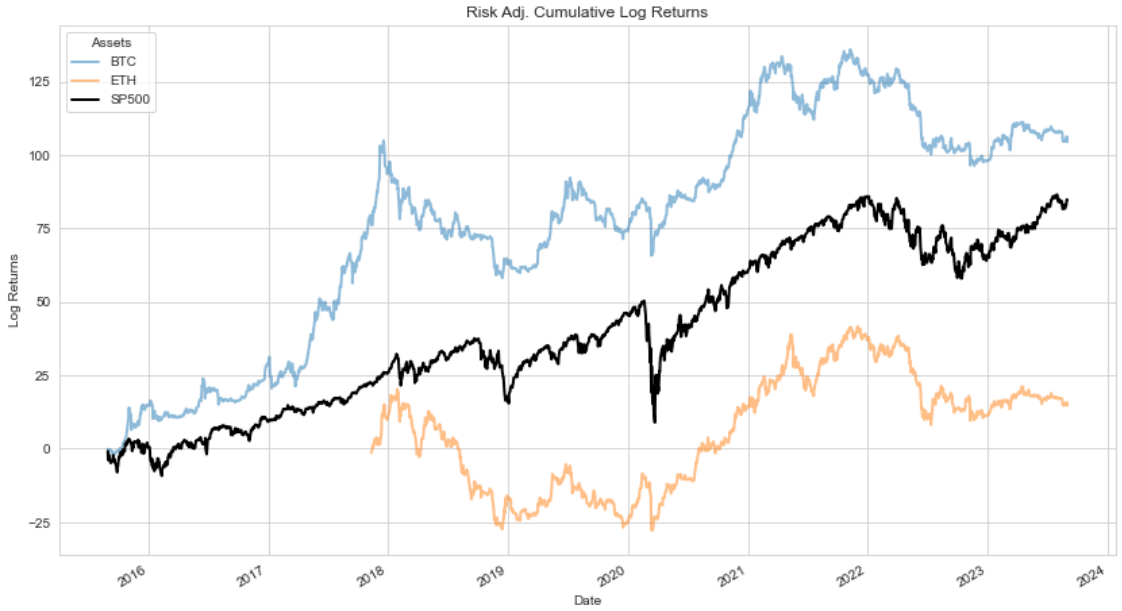
[385]:
```
plt.figure(figsize=(12, 6))

plt.title("Daily Excess Returns",)

BTC['y_actual'].plot(label='BTC Excess Returns', alpha=0.5, linestyle='-',␣
↪linewidth=2)
```

```
ETH['y_actual'].plot(label='ETH Excess Returns', alpha=0.5, linestyle='-',␣
 ↪linewidth=2)

BTC['x'].plot(label='Market Excess Returns', color='black', linestyle='-',␣
 ↪linewidth=2)

plt.legend(loc='upper left', frameon=True, title='Assets')
plt.xlabel("Date")
plt.ylabel("Excess Returns")


plt.tight_layout()
plt.savefig(r"dailyExcessReturns.jpeg", dpi=300, bbox_inches='tight')

plt.show()
```



[395]:
```
plt.figure(figsize=(14,8))
plt.title("Risk Adj. Cumulative Log Returns")
BTC_cumulative = BTC['BTC-USD'].cumsum()
BTC_ra = (BTC['SP500'].std()/BTC['BTC-USD'].std())

ETH_cumulative = ETH['ETH-USD'].cumsum()
ETH_ra = (BTC['SP500'].std()/ETH['ETH-USD'].std())

(BTC_cumulative*BTC_ra).plot(label='BTC', alpha=0.5, linewidth=2)

(ETH_cumulative*ETH_ra).plot(label='ETH', alpha=0.5, linewidth=2)
```

```
# BTC['SP500'].plot(c='black')
BTC['SP500'].cumsum().plot(c='black', linewidth=2)

plt.xlabel("Date")
plt.ylabel("Log Returns")
plt.legend(loc='upper left', frameon=True, title='Assets')
plt.savefig(r"cumulativeLogReturns.jpeg", dpi=300, bbox_inches='tight')

plt.show()
```



[464]:
```
BTC['BTC-USD'].corr(ETH['ETH-USD'])
```

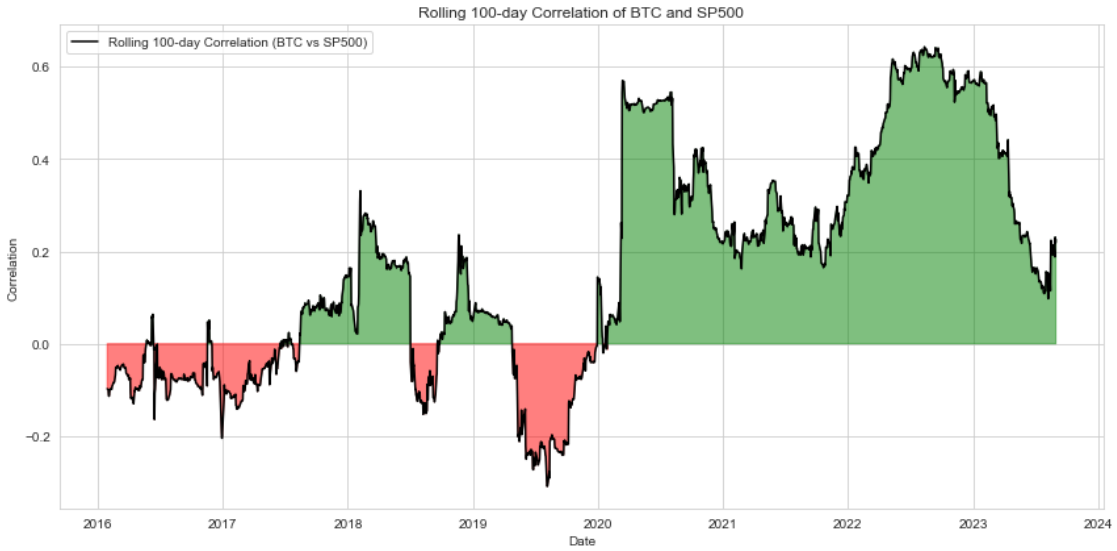
[464]:
```
0.7870153997450345
```

[ ]:
```
BTC['BTC-USD'].corr(BTC['SP500'])
```

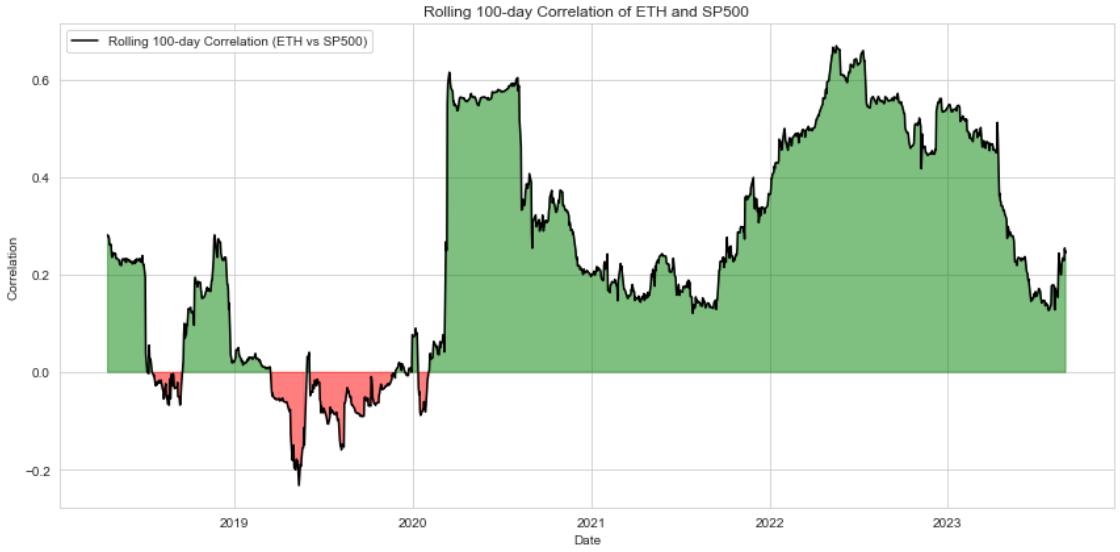
[461]:
```
def plot_rolling_correlation(df, crypto, market, n=100):

    rolling_corr = df[crypto].rolling(n).corr(df[market])

    fig, ax = plt.subplots(figsize=(12, 6))

    above_zero = rolling_corr > 0
    below_zero = rolling_corr <= 0

    ax.plot(rolling_corr, label=f'Rolling {n}-day Correlation ({crypto[0:3]} vs␣
↪{market})', color='k')
```

```
    ax.fill_between(rolling_corr.index, 0, rolling_corr, where=above_zero,␣
↪color='green', alpha=0.5)

    ax.fill_between(rolling_corr.index, 0, rolling_corr, where=below_zero,␣
↪color='red', alpha=0.5)

    ax.set_title(f'Rolling {n}-day Correlation of {crypto[0:3]} and {market}')
    ax.set_xlabel('Date')
    ax.set_ylabel('Correlation')
    ax.legend(loc='upper left')
    plt.tight_layout()
    plt.savefig(f"rollingCorrelation{crypto}.jpeg", dpi=300,␣
↪bbox_inches='tight')

    plt.show()
```

[462]:
```
plot_rolling_correlation(BTC, 'BTC-USD', 'SP500')
```



[463]:
```
plot_rolling_correlation(ETH, 'ETH-USD', 'SP500')
```


Rolling 100-day Correlation of ETH and SP500
Rolling 100-day Correlation (ETH vs SP500)
0.6
0.4
0.2
0.0
−0.2
Correlation
2019
2020
2021
2022
2023
Date