Risk Quantification in Cryptocurrency

DHANUSH

Importance

With the cryptocurrency market soaring to a nearly $1.5 trillion in market capitalization, the allure of these digital assets is unmistakable. However, amidst this financial exuberance lies a palpable concern—the inherent volatility of these unbacked tokens, exemplified by stalwarts like Bitcoin. In a landscape where market dynamics can swiftly transform, the need to gauge and mitigate risks becomes paramount

Cryptocurrency, often hailed as a financial frontier, operates in a unique space where traditional valuation metrics hold less sway. The absence of underlying assets backing popular coins amplifies their susceptibility to extreme fluctuations. This volatility, coupled with the substantial market cap, poses challenges that demand a nuanced approach to risk assessment.

For this purpose, we calculate VaR, which is one of the most widely used metric to measure the control and level of risk exposure.

.

Calculating VaR

As it necessary for us to calculate say probability for a certain extent of loss with respect to mean in say terms of standard deviation. We may do the following.

$$P(X < Z) = \textbf{Required}\ \%.$$

For example, P(X<-1.645)= 5%, and for a normal distribution concentrated about 0, we have a 5% chance to lose more than 1.645 standard deviations.

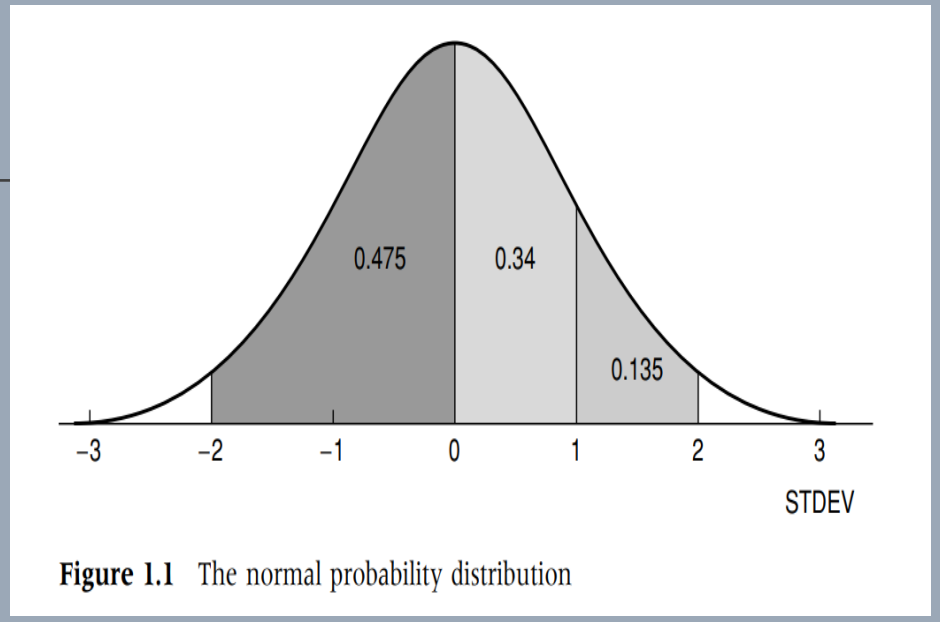


Figure 1.1 The normal probability distribution

Let us consider a more general-purpose equation based on the definition, to account for the non-normal distributions we may encounter.

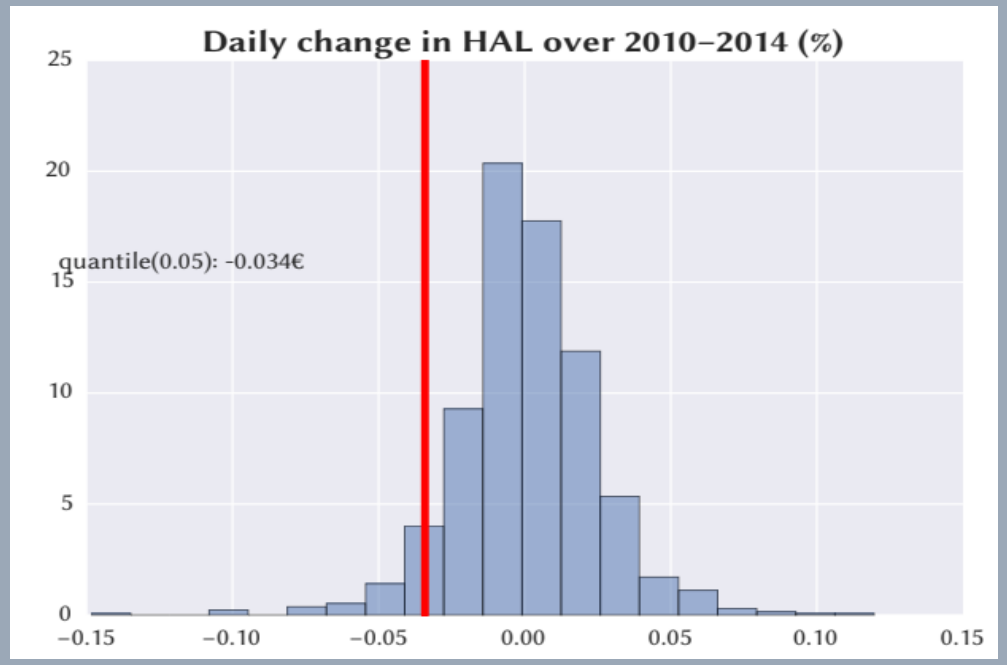
$$p = Pr[L(h) \geq VaR] = 1 - Pr[L(h) < VaR]$$

$$\textbf{X day VaR} = \sqrt{\text{X}} * (\textbf{daily VaR})$$

Methods of Calculation

- Historical Bootstrap Method
- Variance – Covariance Method
- Geometric Brownian Motion and Monte Carlo
- GARCH

Historical Bootstrap Method



Functions under the hypothesis that history is a representative of future activity. We calculate empirical quantiles from a histogram of daily returns and take the sought confidence interval.

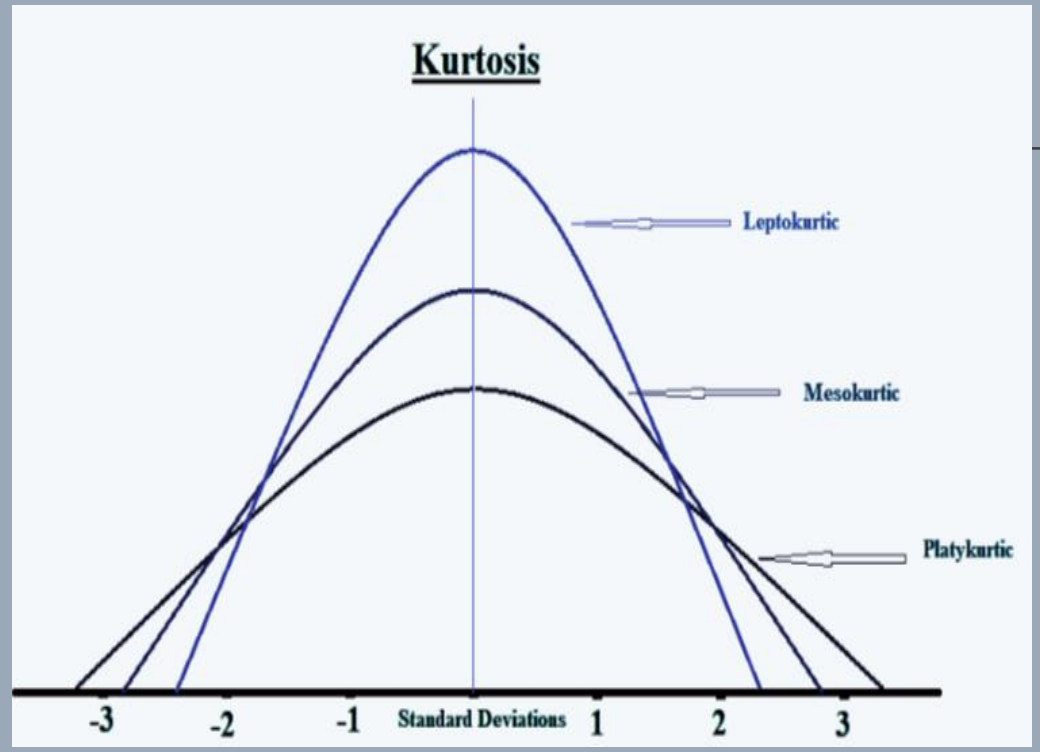
If we were to use this hypothesis on this model, we may say:

a) Our daily loss won't exceed
   3.4% with 95% certainty.

b) For a 1M$ investment: one day 5%VaR = 0.034*1M$ = 34K$

(Here, the 0.05 quantile indicates the 5^{th} percentile)

Variance Covariance Method



This method operates under the assumption that we're dealing with a normal distribution.

However, in our analysis we discover our bitcoin distribution to be **Leptokurtic**, with positive excess kurtosis. So, we will not explore this in depth.

VaR calculation for a single asset is straightforward. From the distribution of returns calculated from daily price series, the standard deviation (σ) under a certain time horizon is estimated. The daily VaR is simply a function of the standard deviation and the desired confidence level and can be expressed as:

$$\text{VaR} = \alpha \cdot \sigma$$

Where the parameter α links the quantile of the normal distribution and the standard deviation: α = 2.33 for p = 99% and α = 1.645 for p = 90%.

Geometric Brownian Motion

Let us explore, how this model was fabricated from the root assumption of Markov Property.

Markov Property

Wiener Process

In accordance with the efficient market hypothesis, it states the **present price of a stock impounds all the information contained in a record of past prices**.

This describes a variable Δz, whose change is measured over the interval Δt such that its mean change is zero and variance is proportional to Δt:

$$\Delta z \sim N(0, \Delta t) if\ \varepsilon \sim N(0,1), we\ may\ say, \Delta z = \epsilon\sqrt{\Delta t}$$

More formally, a wiener process is type of Markov stochastic process with a mean change of 0 and a variance rate of 1.0 per year. The same equation is used to model the motion of a particle subjected to a large number of small molecular shocks **(Brownian Motion).**

Wiener Process (GBM cont.)

Consider the increase in the value of z during a relatively long period of time, T. This can be denoted by $z(T) - z(0)$. It can be regarded as the sum of the increases in z in N small time intervals of length δt, where

$$N = \frac{T}{\delta t}$$

Thus,

$$z(T) - z(0) = \sum_{i=1}^{N} \epsilon_i \sqrt{\delta t} \tag{11.2}$$

where the ϵ_i $(i = 1, 2, \ldots, N)$ are random drawings from $\phi(0, 1)$. From second property of Wiener processes, the ϵ_i's are independent of each other. It follows from equation (11.2) that $z(T) - z(0)$ is normally distributed with

$$\text{mean of } [z(T) - z(0)] = 0$$

$$\text{variance of } [z(T) - z(0)] = N\,\delta t = T$$

$$\text{standard deviation of } [z(T) - z(0)] = \sqrt{T}$$

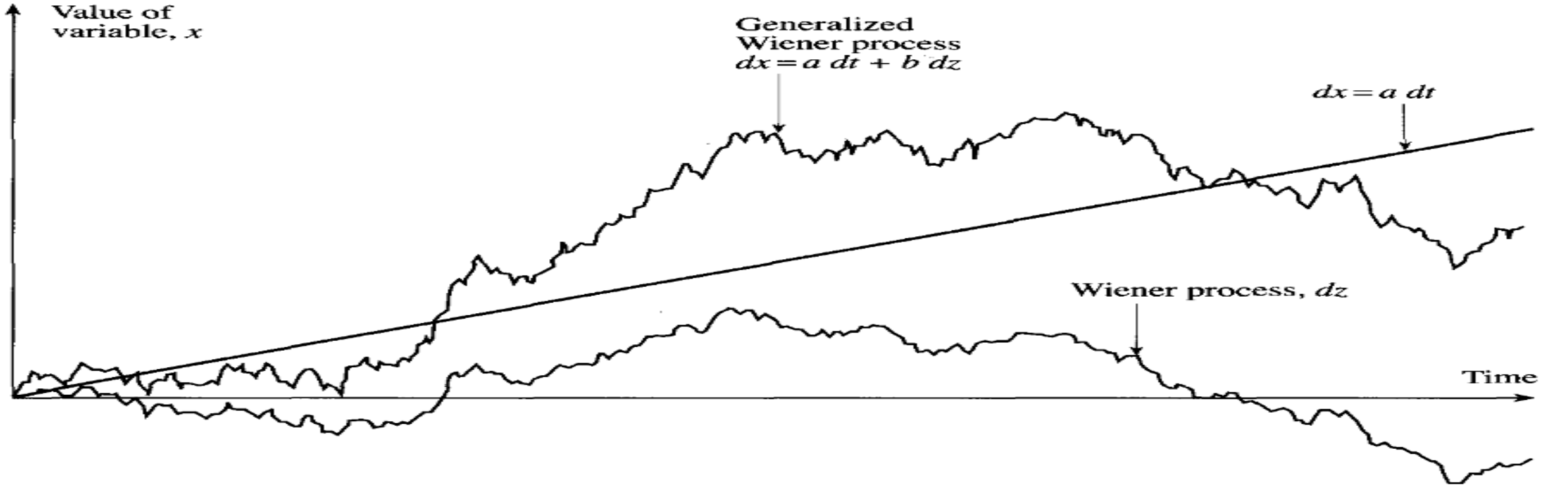
Generalized Wiener Process (GBM cont.)

In practice, we make modifications to the above equation to account for a potential "drift" with time for the variable. So, we make the following modification:

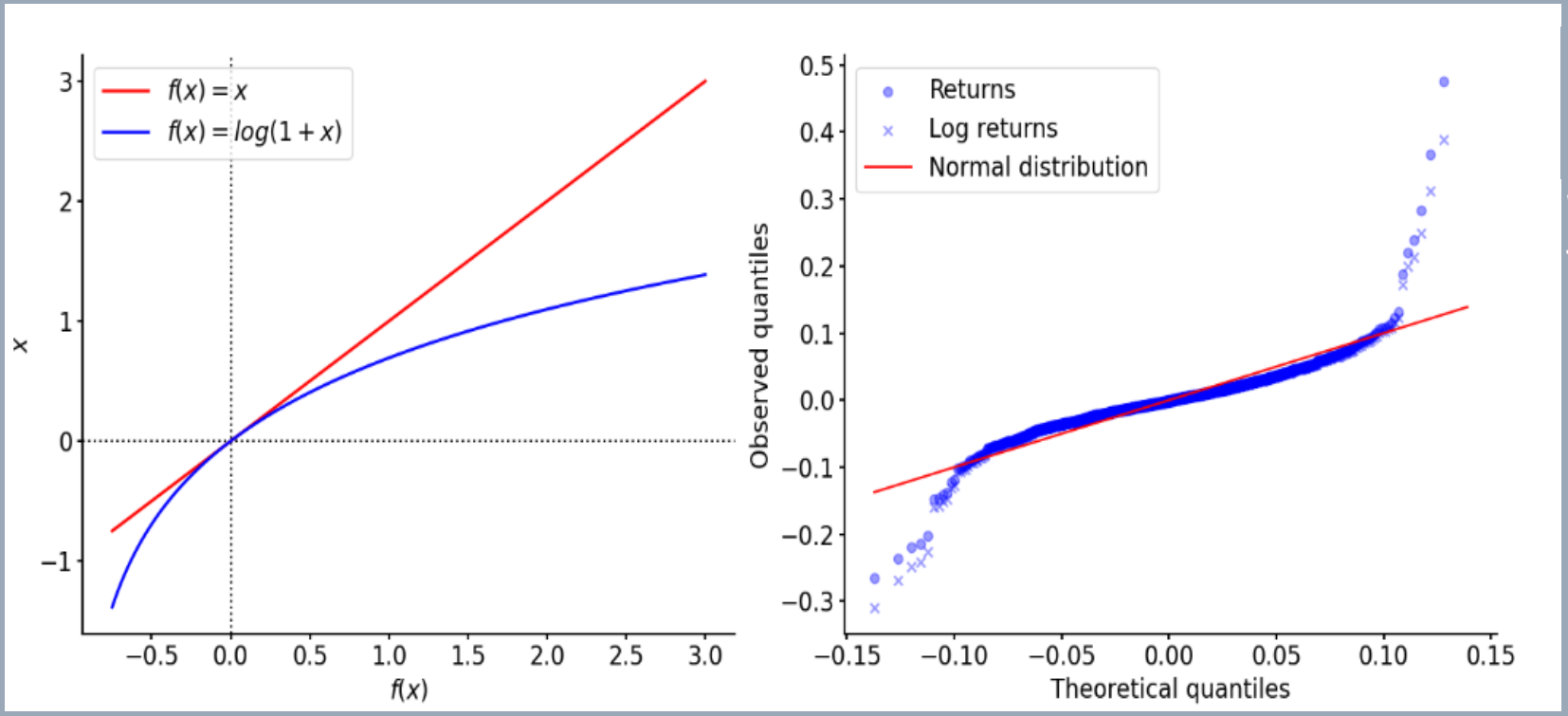
$$\partial x = a\partial t + b\varepsilon\sqrt{\delta t}$$

For any variable x :

- a **denotes drift coefficient,** $a\partial t =$ **drift.**
- $b\varepsilon\sqrt{\delta t}$ **represents volatility or noise.**



Log Returns (GBM cont.)



We use log returns over returns. To handle the large upward movements and ensure price remains positive.

Return percentage (GBM cont.)

Now, a key principle that must be introduced into these equations, are that investors generally seek constant investment return percentage (rather than absolute returns itself). So, we consider a modified variable for our Markov Process to get the following equation:

$$\frac{\Delta S}{S} = \mu\Delta t + \sigma\sqrt{\Delta t} \;\; ; \;\; \left(\frac{\Delta S}{S} \sim N\left(\mu\Delta t, \sigma\sqrt{\Delta t}\right)\right)$$

An important modification when we are considering logarithms of the parameters is:

$$\ln S_T - \ln S_0 \sim \phi\left[\left(\mu - \frac{\sigma^2}{2}\right)T,\ \sigma\sqrt{T}\right]$$

$$\ln S_T \sim \phi\left[\ln S_0 + \left(\mu - \frac{\sigma^2}{2}\right)T,\ \sigma\sqrt{T}\right]$$

GARCH Model

GARCH(p,q) general model:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \cdot \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \cdot \sigma_{t-j}^2$$

Here,

- σ_t^2 is the conditional variance at time t,
- ω is the model constant,
- α_i are coefficients determining the impact of past squared shocks,
- ε_{t-i}^2 is the squared return at time $t-i$,
- β_j are coefficients determining the impact of past conditional variances, and
- σ_{t-j}^2 is the conditional variance at time $t-j$.

Why GARCH?

Volatility Clustering

Dynamic Adaptation

Memory Effects

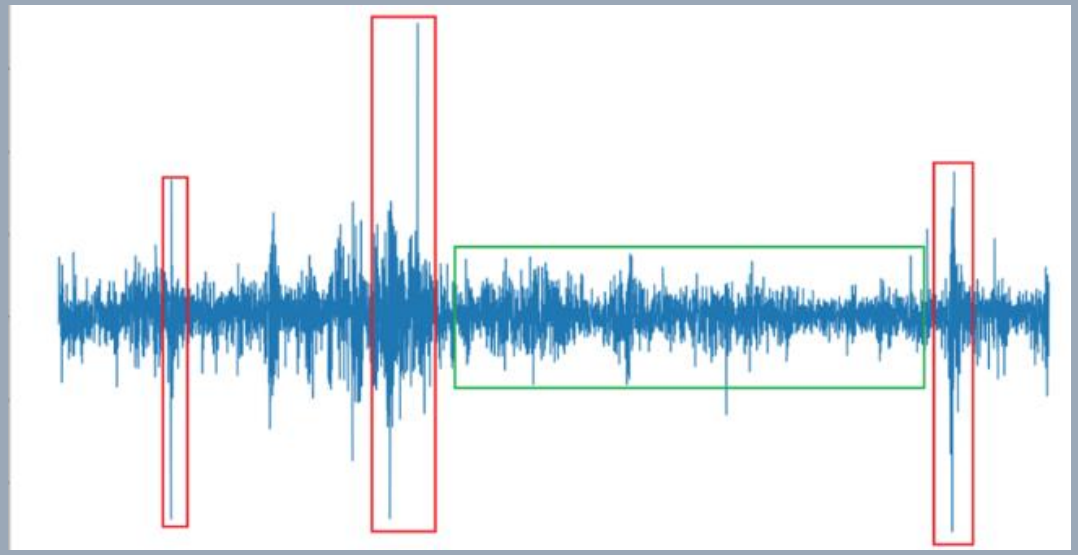
Volatility clustering

Volatility clustering refers to the empirical observation that *periods of high volatility tend to be followed by more periods of high volatility, and vice versa*. This phenomenon is inherent in financial markets, reflecting the dynamic nature of investor sentiment and external market shocks

GARCH models are designed to capture this clustering effect by incorporating lagged terms that account for past volatility. The model recognizes that volatility tends to persist, allowing it to accurately reflect the clustering pattern observed in financial time series data

$\alpha . \varepsilon^2{}_{t-1}$ is the autoregressive component that contributes to volatility persistence, representing the impact of past squared shocks on volatility.

(in the GARCH(1,1) equation: $\sigma_t^2 = \omega + \alpha . \varepsilon^2{}_{t-1} + \beta . \sigma_{t-1}^2$)



- **Volatility Clustering:** AR, MA, ARMA, and ARIMA models fail to effectively capture volatility clustering in financial time series data.
- **ARCH Model:** ARCH successfully explains volatility clustering but lacks a moving average component.
- **GARCH Model:** GARCH addresses the limitations of ARCH by introducing a moving average component, making it adept at capturing both persistence and adaptability in volatility.

In essence, while AR, MA, ARMA, and ARIMA models focus on mean-reversion and unexpected events, ARCH and GARCH models specifically target volatility clustering by incorporating autoregressive processes in the variance. GARCH, with its additional moving average component, emerges as a more comprehensive solution, successfully capturing the intricacies of volatility dynamics in financial markets.

Dynamic adaptation

Financial markets are dynamic and subject to ever-changing conditions. Static models, which assume constant volatility, may inadequately represent the true nature of market dynamics. Dynamic adaptation is crucial for capturing shifts in volatility over time.

GARCH models dynamically adapt to changing volatility conditions by updating their estimates based on recent information. The inclusion of both autoregressive and moving average components allows the model to flexibly respond to evolving market conditions.

The moving average component $\beta . \sigma_{t-1}^2$ in the GARCH(1,1) model accounts for the impact of past conditional variances on current volatility, contributing to the model's adaptability.

(in the GARCH(1,1) equation: $\sigma_t^2 = \omega + \alpha . {\varepsilon^2}_{t-1} + \beta . \sigma_{t-1}^2$)

Memory effects

Memory effects in volatility describe the persistence of past shocks influencing future volatility. Investors often make decisions based on historical information, and the inclusion of memory effects is crucial for modeling the lingering impact of past events.

The lagged terms in GARCH models allow for the incorporation of memory effects. The model considers not only the immediate impact of shocks but also their lasting influence on subsequent volatility.

The autoregressive component $\alpha.{\varepsilon^2}_{t-1}$ explicitly accounts for the squared shocks at time $t-1$, representing the memory effect in the GARCH(1,1) model.

(in the GARCH(1,1) equation: $\sigma_t^2 = \omega + \alpha.{\varepsilon^2}_{t-1} + \beta.\sigma_{t-1}^2$)

Code

I've taken the data for bitcoin prices from coingecko api, from 2013 to 2022, and have particularly performed back-testing in the last 400 days of the period since it is widely reported to have been a high-volatility period.

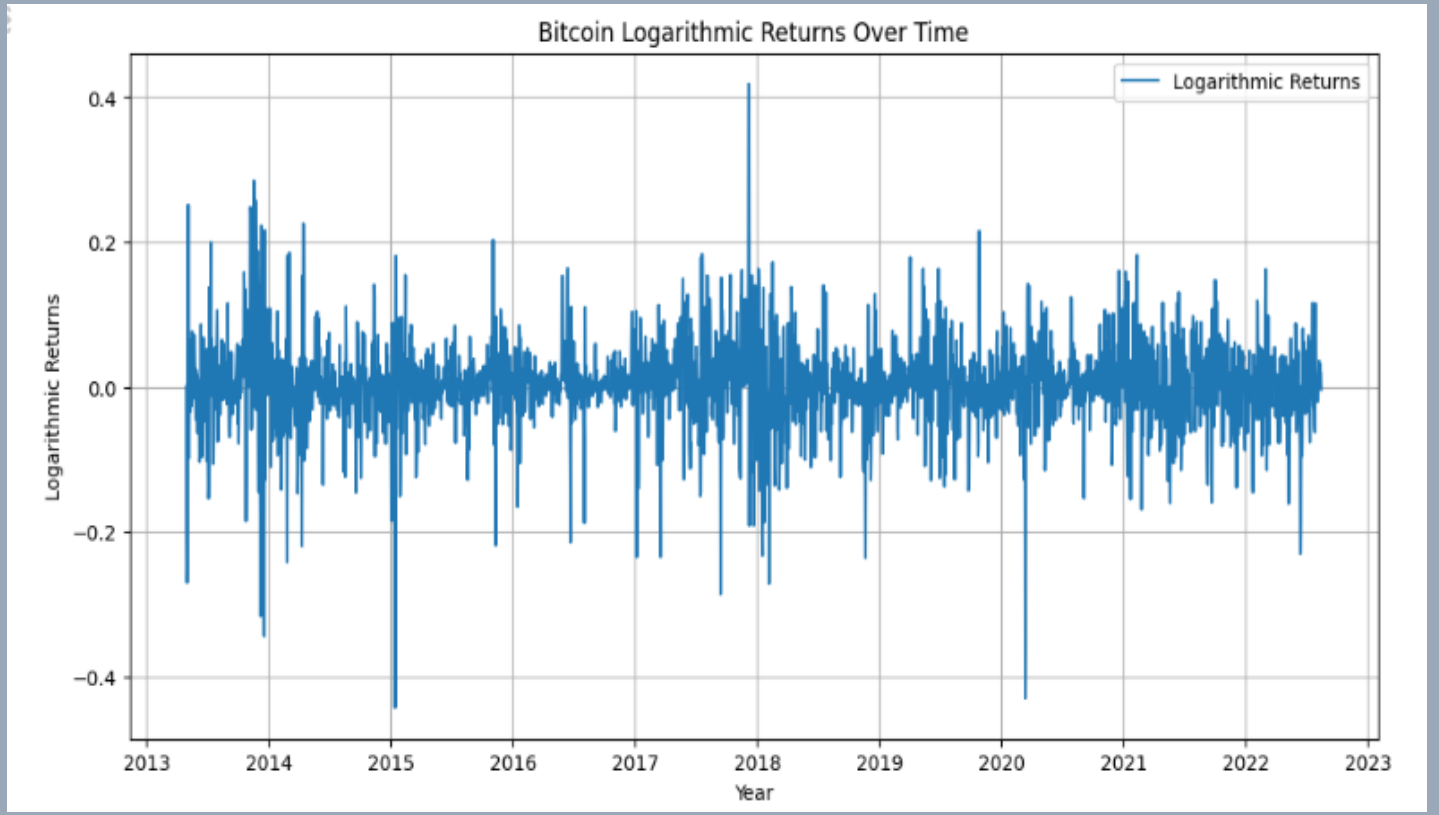
Sending requests

```python
import requests

url = "https://api.coingecko.com/api/v3/coins/bitcoin/market_chart/range"


# GOT THE ABOVE URL REQUEST AT PUBLIC API DOCUMENTATION OF COINGECKO

params = {
    'vs_currency': 'usd',
    'from': 1367107200,  # Start timestamp in seconds
    'to': 1660521600,    # End timestamp in seconds
    'precision': 0
}

response = requests.get(url, params=params)
data = response.json()

closing_prices = [price[1] for price in data['prices']]

print("Closing prices:")
for timestamp, price in zip(data['prices'], closing_prices):
    print(f"{timestamp[0]}: {price} USD")
```

```python
import numpy as np
# Calculate logarithmic returns
log_returns = [np.log(price) - np.log(closing_prices[i - 1]) for i, price in enumerate(closing_prices[1:])]

print("\nLogarithmic returns:")
for i, log_return in enumerate(log_returns):
    date = timestamp[0] / 1000  # Timestamp of the last closing price
    print(f"{date}: {log_return}")

# Store logarithmic returns in a list

# You can use 'log_returns' for further analysis or storage
print("\nStored logarithmic returns:")
print(log_returns)
```

Bitcoin returns

```
Bitcoin Statistics:
Minimum: -0.4434925036974038
Maximum: 0.41798159715311733
1st Quartile: -0.022090957814191903
3rd Quartile: 0.029594191911304 08
Mean: 0.0030478619256193184
Median: 0.003428454225915 0896
Sum: 10.341395513626347
Variance: 0.003392829758358 0547
Stdev: 0.05824800218340587
Skewness: -0.3944992110858868
Kurtosis: 5.942624452458849
```



Codes for VaR calculation

```python
# Function to calculate Historical VaR
def historical_var(log_returns, P0):
    var = np.percentile(log_returns, 5)
    return var * P0
```

Historical VaR

```python
# Function to calculate GBM VaR
def GBM_var(log_returns, P0):
    N = 365  # Number of time steps
    mu = np.mean(log_returns)
    sigma = np.std(log_returns)
    Wt = np.cumsum(np.random.normal(0, 1, N))
    t = np.arange(1, N + 1) / 365
    drift = (mu - 0.5 * sigma**2) * t
    diff = sigma * Wt
    Pt = P0 * np.exp(drift + diff)
    log_returns_simulations = [np.log(Pt[1:] / Pt[:-1]) for Pt in bitcoin_paths]
    var_1_day_log_returns = np.percentile(np.concatenate(log_returns_simulations), 95)
    var_1_day = P0 * np.exp(var_1_day_log_returns)
    return var_1_day
```

GBM VaR

Codes for VaR calculation

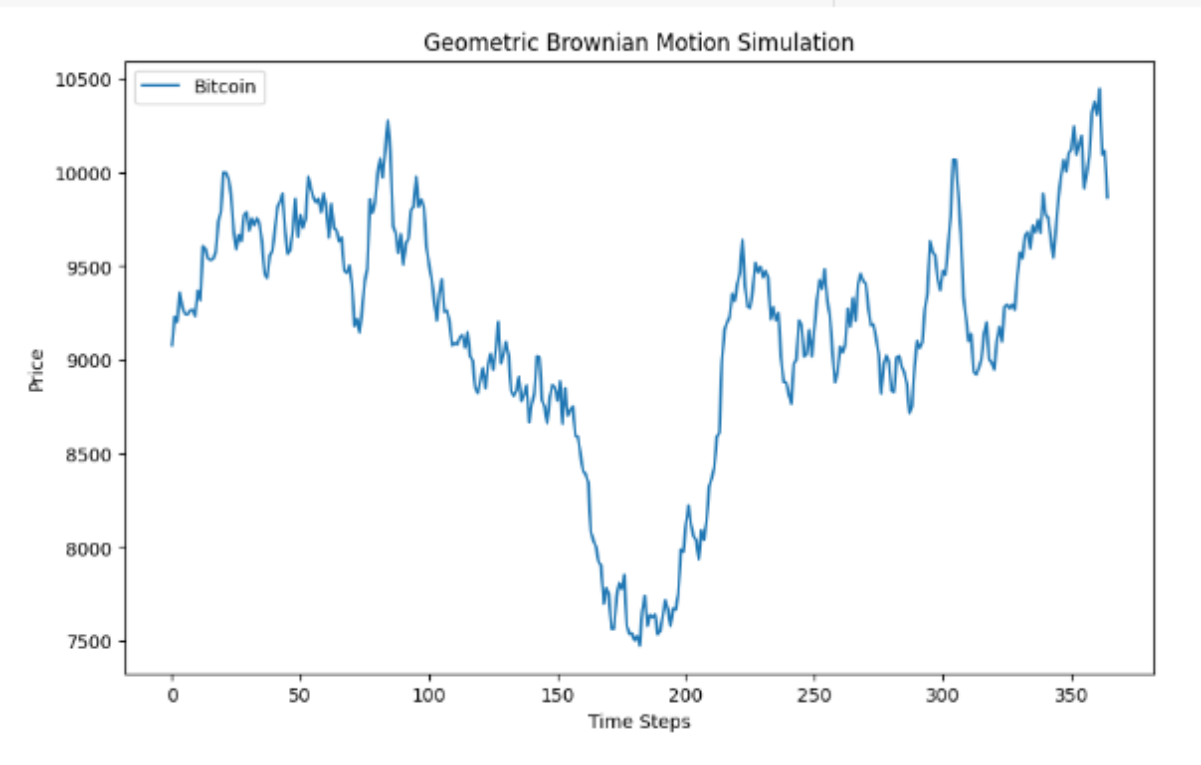
```python
# Function to calculate GARCH VaR
def GARCH_var(log_returns, P0):
    model = arch_model(log_returns, vol='Garch', p=1, q=1)
    result = model.fit(disp='off')
    daily_volatility_forecast = result.conditional_volatility[-1]
    bitcoin_var_1day = daily_volatility_forecast * norm.ppf(0.95)
    bitcoin_var_1day_dollars = bitcoin_var_1day * P0
    return bitcoin_var_1day_dollars
```

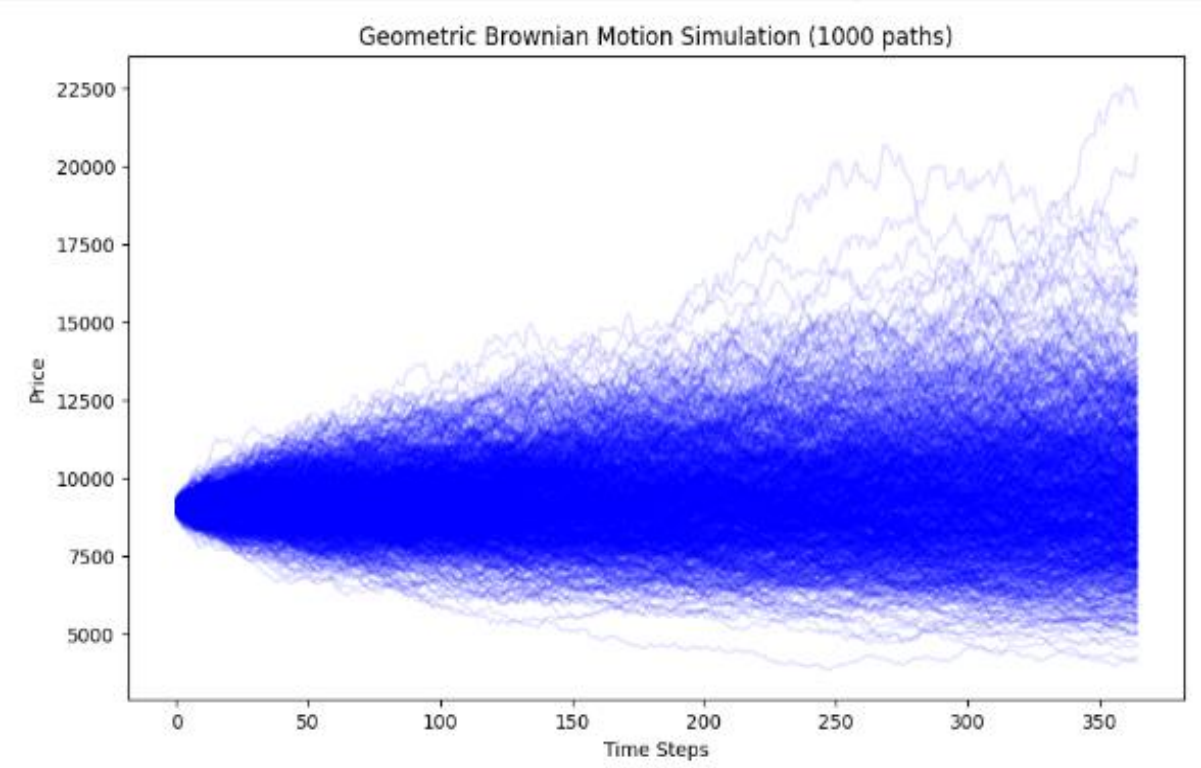
GARCH VaR

```python
# Function to calculate using Monte Carlo Simulations
def MC_var(log_returns, P0, num_simulations=1000):
    N = 365  # Number of time steps
    mu = np.mean(log_returns)
    sigma = np.std(log_returns)
    simulations = []
    for _ in range(num_simulations):
        Wt = np.cumsum(np.random.normal(0, 1, N))
        t = np.arange(1, N + 1) / 365
        drift = (mu - 0.5 * sigma**2) * t
        diff = sigma * Wt
        Pt = P0 * np.exp(drift + diff)
        simulations.append(Pt)
    log_returns_simulations = [np.log(Pt[1:] / Pt[:-1]) for Pt in simulations]
    var_1_day_log_returns = np.percentile(np.concatenate(log_returns_simulations), 95)
    var_1_day = P0 * np.exp(var_1_day_log_returns)
    return var_1_day
```

Monte Carlo VaR

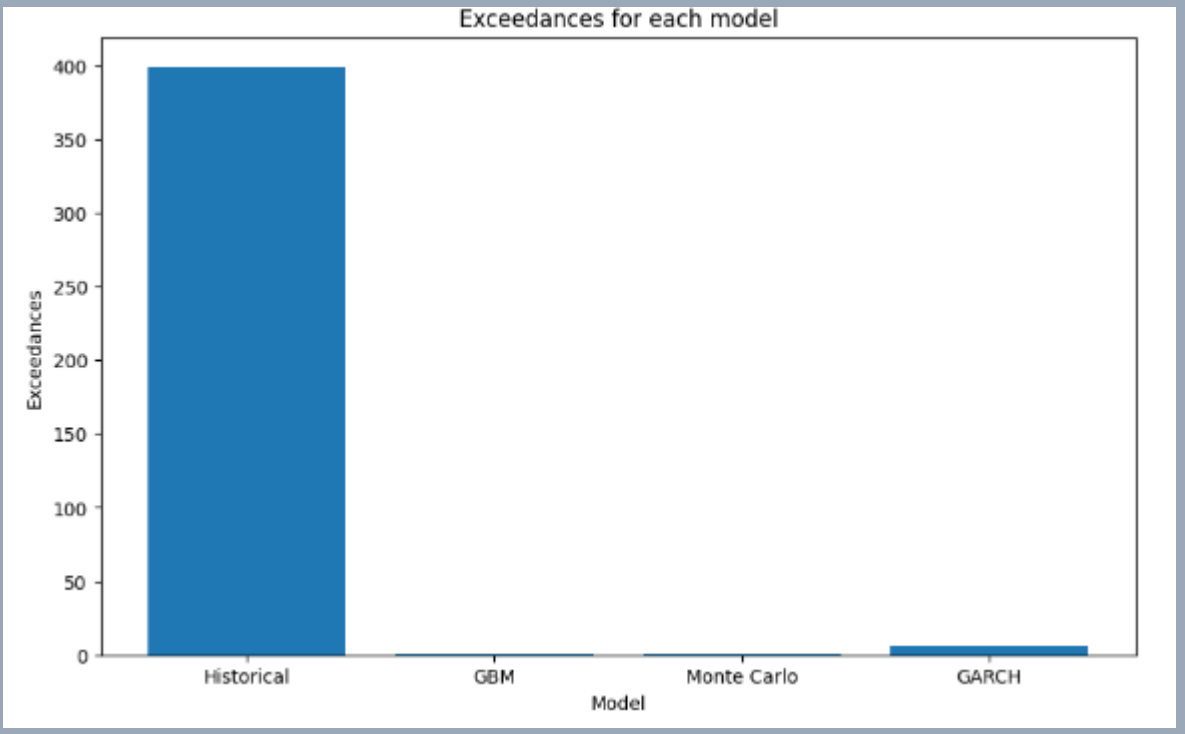
Results

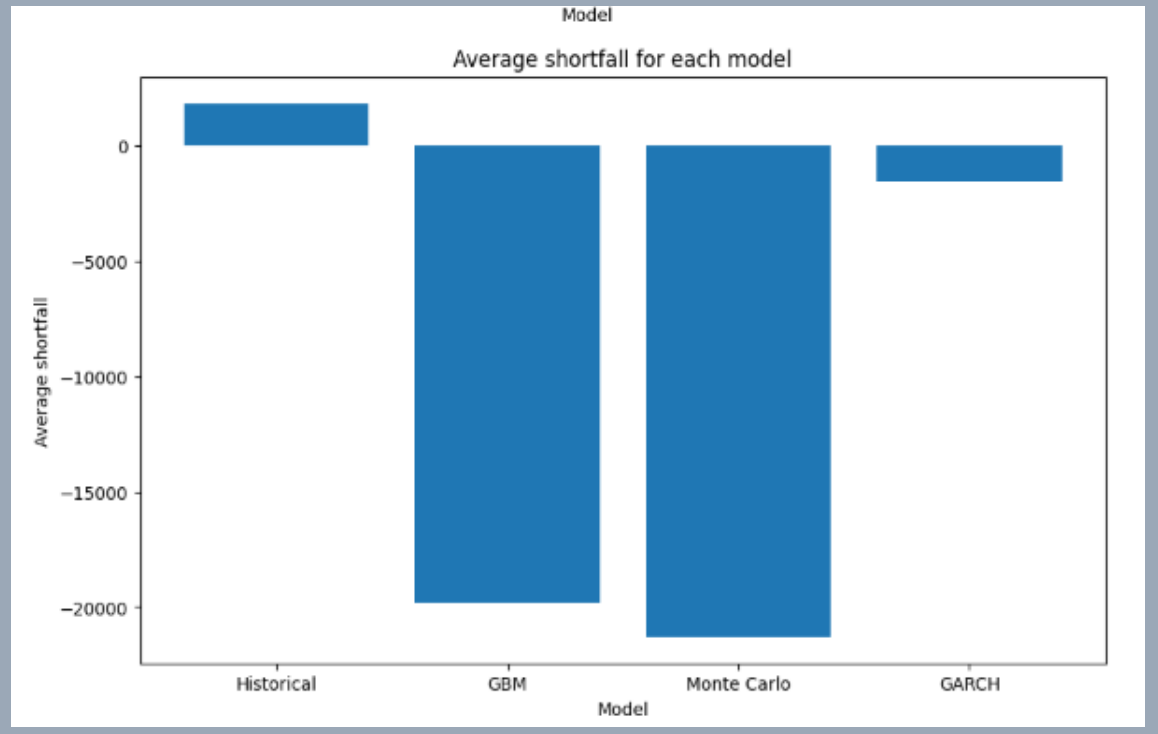

Geometric Brownian Motion Simulation
Bitcoin
Price
Time Steps
10500
10000
9500
9000
8500
8000
7500
0
50
100
150
200
250
300
350


Geometric Brownian Motion Simulation (1000 paths)
Price
Time Steps
22500
20000
17500
15000
12500
10000
7500
5000
0
50
100
150
200
250
300
350

Results

Exceedances and Shortfalls for each model:





Exceedances

Historical VaR exceedances: 399 out of 400 days (99.75%)

GBM VaR exceedances: 1 out of 400 days (0.25%)

Monte Carlo VaR exceedances: 1 out of 400 days (0.25%)

GARCH VaR exceedances: 6 out of 400 days (1.50%)

Avg. Shortfall

Historical VaR : 1819.1861349530495

GBM : -19830.872627019628

Monte Carlo : -21280.0494621295

GARCH(1,1) : -1541.467217966031

Results

The shortfall indicates the “real value” – “estimated value” : The fact that historical VaR produced a positive shortfall is indicative of the fact that Historical VaR has consistently downplayed the risk involved and similarly GBM, and Monte Carlo have consistently kept exceedances at bay but at the cost of a severe overplay of the risk involved which may be seen from the shortfall list

The results agree with our estimate, that GARCH would work out best at modelling the same due to its ability to consider heteroskedastic properties.