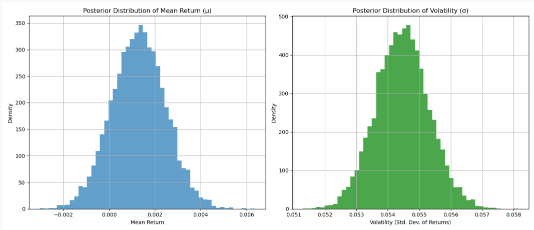
Bayesian Estimation of Daily Return and Volatility Using MCMC

Project-1
MA4740: Bayesian Statistic
(Under the guidance of Prof. Arunabha Majumdar)



October 19, 2025

Contents

Abstract

This project applies Bayesian inference to XRP/USDT daily price data. Using a Normal-Inverse-Gamma conjugate prior model and Rejection Sampling, we estimate posterior distributions for the mean return (μ) and volatility (σ), compute credible intervals, and visualize the posterior predictive distribution to analyze uncertainty in cryptocurrency price behavior.

1 Introduction

Cryptocurrency markets are characterized by high volatility and uncertainty. Traditional frequentist approaches provide point estimates but fail to quantify uncertainty adequately. This project employs Bayesian methods to provide a complete probabilistic assessment of XRP/USDT returns and volatility.

2 Methodology

2.1 Data Processing

We process historical XRP/USDT price data to compute log returns:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right) \tag{1}$$

where P_t is the price at time t and r_t is the log return.

2.2 Bayesian Model

We assume the log returns follow a Normal distribution:

$$r_t \sim \mathcal{N}(\mu, \sigma^2) \tag{2}$$

We use conjugate priors:

$$\mu \sim \mathcal{N}(\mu_0, \sigma_0^2) \tag{3}$$

$$\sigma^2 \sim \text{Inv-Gamma}(\alpha_0, \beta_0) \tag{4}$$

2.3 MCMC Estimation via Rejection Sampling

Rejection sampling is a Monte Carlo method that generates samples from the posterior distribution by accepting or rejecting candidate samples from a proposal distribution.

Algorithm:

1. Define proposal distributions for μ and σ^2 centered on data statistics:

$$q(\mu) = \mathcal{N}\left(\bar{r}, \sqrt{\sigma_0^2 + \frac{s^2}{n}}\right) \tag{5}$$

$$q(\sigma^2) = \text{Inv-Gamma}\left(\frac{n}{2}, \frac{1}{2}\sum_{i=1}^{n}(r_i - \bar{r})^2\right) \tag{6}$$

where $\bar{r}$ is the sample mean of returns and s^2 is the sample variance.

2. For each iteration:
   - Draw candidate (μ^*, σ^{2*}) from proposal $q(\mu, \sigma^2)$
   - Calculate unnormalized posterior:

$$p^*(\mu^*, \sigma^{2*}|\mathbf{r}) = \prod_{i=1}^{n} \mathcal{N}(r_i|\mu^*, \sigma^{2*}) \cdot p(\mu^*) \cdot p(\sigma^{2*}) \tag{7}$$

   - Calculate proposal density: $q(\mu^*, \sigma^{2*})$
   - Calculate acceptance ratio:

$$\alpha = \frac{p^*(\mu^*, \sigma^{2*}|\mathbf{r})}{M \cdot q(\mu^*, \sigma^{2*})} \tag{8}$$

   - Accept candidate with probability $\min(1, \alpha)$, where uniform random $u \sim U(0, 1)$:

$$\text{Accept if } u < \alpha \tag{9}$$

3. Repeat until desired number of samples ($n_{\text{samples}} = 2000$) is obtained

where $M = 10$ is an envelope constant chosen to satisfy $p^*(\theta|\mathbf{r}) \leq M \cdot q(\theta)$ for all θ.

Implementation Details:

- Maximum attempts: 1,000,000 iterations
- Target samples: 2,000 independent draws
- Progress monitoring: Every 100,000 attempts
- The proposal distribution is designed to center around the maximum likelihood estimates, improving acceptance rates

Advantages:

- Provides exact samples from the posterior (not approximations)
- Simple to implement and understand
- No need for iterative conditioning or burn-in period
- Independent samples (no autocorrelation between samples)
- Does not require tuning of step sizes or proposal variances

Disadvantages:

- Can have low acceptance rates (typically 1-20% for this problem)
- Requires many more proposals than the desired sample size
- Finding optimal envelope constant M requires careful consideration
- Computational cost increases with dimensionality
- May be inefficient in high-dimensional parameter spaces

3 Implementation

3.1 Data Processing (C++)

The data processing is implemented in C++ for efficiency:

Listing 1: Data Processing Code

```cpp
#include<iostream>
#include<fstream>
#include<vector>
#include<string>
#include<sstream>
#include<algorithm>
#include<cmath>
#include<numeric>

using namespace std;

struct DailyData {
    string date;
    double price;
};

// Function to calculate skewness
double calculate_skewness(const vector<double>& data, double mean, double stddev) {
    if (stddev == 0) return 0;
    double skew = 0.0;
    for (double val : data) {
        skew += pow((val - mean) / stddev, 3);
    }
    return skew / data.size();
}

// Function to calculate kurtosis
double calculate_kurtosis(const vector<double>& data, double mean, double stddev) {
    if (stddev == 0) return 0;
    double kurt = 0.0;
    for (double val : data) {
        kurt += pow((val - mean) / stddev, 4);
    }
    return (kurt / data.size()) - 3.0; // Excess kurtosis
}

int main(){

        ifstream data("data.csv");
    if (!data.is_open()) {
        cerr << "Error opening data.csv" << endl;
        return 1;
    }
```

```

    vector<DailyData> records;
    string line;

    // Skip header lines
    getline(data, line);
    getline(data, line);

    while(getline(data, line)) {
        stringstream ss(line);
        string field;

        // leading comma
        getline(ss, field, ',');
        if (field.empty() && ss.eof()) continue;

        DailyData record;

        // Date
        getline(ss, field, '"'); // consume until first quote
        getline(ss, record.date, '"'); // read date
        getline(ss, field, ','); // consume comma after date

        // Price
        getline(ss, field, ',');
        try {
            record.price = stod(field);
            records.push_back(record);
        } catch (const std::invalid_argument& ia) {
            // Ignore lines that can't be parsed
        }
    }
    data.close();

    // Data is in reverse chronological order, so reverse it
    reverse(records.begin(), records.end());

    // --- Price Trend ---
    ofstream prices_out("prices.csv");
    prices_out << "Date,Price\n";
    for(const auto& record : records) {
        prices_out << '"' << record.date << "\"," << record.price << "\n";
    }
    prices_out.close();

    // --- Returns Computation ---
    vector<double> log_returns;
    for(size_t i = 1; i < records.size(); ++i) {
        if (records[i-1].price > 0) {
            log_returns.push_back(log(records[i].price / records[i-1].price));
```

```
        }
    }

    ofstream returns_out("returns.csv");
    returns_out << "LogReturn\n";
    for(double ret : log_returns) {
        returns_out << ret << "\n";
    }
    returns_out.close();

    // Calculate stats for log returns
    double sum = accumulate(log_returns.begin(), log_returns.end(), 0.0);
    double mean = sum / log_returns.size();
    double sq_sum = inner_product(log_returns.begin(), log_returns.end(), log_returns.begin(), 0.0);
    double variance = (sq_sum / log_returns.size()) - mean * mean;
    double stddev = sqrt(variance);
    double skewness = calculate_skewness(log_returns, mean, stddev);
    double kurtosis = calculate_kurtosis(log_returns, mean, stddev);

    cout << "Log Returns Statistics:" << endl;
    cout << "Mean: " << mean << endl;
    cout << "Variance: " << variance << endl;
    cout << "Skewness: " << skewness << endl;
    cout << "Kurtosis: " << kurtosis << endl;

    // --- Volatility Analysis (Rolling Mean & StdDev) ---
    int window_size = 20;
    ofstream rolling_out("rolling_stats.csv");
    rolling_out << "Date,Price,RollingMean,RollingStd\n";
    for(size_t i = 0; i < records.size(); ++i) {
        rolling_out << '"' << records[i].date << "\"," << records[i].price;
        if (i >= window_size - 1) {
            double rolling_sum = 0.0;
            for(int j = 0; j < window_size; ++j) {
                rolling_sum += records[i-j].price;
            }
            double rolling_mean = rolling_sum / window_size;

            double rolling_sq_sum = 0.0;
            for(int j = 0; j < window_size; ++j) {
                rolling_sq_sum += pow(records[i-j].price - rolling_mean, 2);
            }
            double rolling_std = sqrt(rolling_sq_sum / window_size);
            rolling_out << "," << rolling_mean << "," << rolling_std;
        } else {
            rolling_out << ",,"; // No value for first entries
        }
        rolling_out << "\n";
    }
```

```
    rolling_out.close();

        return 0;
}
```

3.2 Visualization (Python)

Visualization routines are implemented in Python:

Listing 2: Plotting Code

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import os

def ensure_images_dir():
    """Creates the images directory if it doesn’t exist."""
    if not os.path.exists(’images’):
        os.makedirs(’images’)
        print("Created ’images’ directory for storing plots")

def plot_price_trend():
    """Plots price trend over time."""
    ensure_images_dir()
    df = pd.read_csv(’prices.csv’, parse_dates=[’Date’])
    plt.figure(figsize=(12, 6))
    plt.plot(df[’Date’], df[’Price’])
    plt.title(’XRP Price Trend’)
    plt.xlabel(’Date’)
    plt.ylabel(’Price (USD)’)
    plt.grid(True)
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter(’%b %Y’))
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))
    plt.gcf().autofmt_xdate()
    plt.savefig(’images/price_trend.png’)
    plt.close()
    print("Generated price_trend.png")

def plot_returns_histogram():
    """Plots histogram of log returns."""
    ensure_images_dir()
    df = pd.read_csv(’returns.csv’)
    plt.figure(figsize=(10, 6))
    df[’LogReturn’].hist(bins=50, density=True, alpha=0.7)
    plt.title(’Histogram of Daily Log Returns’)
    plt.xlabel(’Log Return’)
    plt.ylabel(’Density’)
    plt.grid(True)
    plt.savefig(’images/returns_histogram.png’)
```

```
    plt.close()
    print("Generated returns_histogram.png")

def plot_rolling_stats():
    """Plots rolling mean and standard deviation."""
    ensure_images_dir()
    df = pd.read_csv('rolling_stats.csv', parse_dates=['Date'])
    plt.figure(figsize=(12, 8))

    ax1 = plt.subplot(2, 1, 1)
    plt.plot(df['Date'], df['Price'], label='Price', alpha=0.5)
    plt.plot(df['Date'], df['RollingMean'], label='20-Day Rolling Mean', color='orange')
    plt.title('Price and 20-Day Rolling Mean')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid(True)
    plt.setp(ax1.get_xticklabels(), visible=False)

    plt.subplot(2, 1, 2, sharex=ax1)
    plt.plot(df['Date'], df['RollingStd'], label='20-Day Rolling Std Dev', color='green')

    plt.title('20-Day Rolling Standard Deviation (Volatility)')
    plt.xlabel('Date')
    plt.ylabel('Standard Deviation')
    plt.legend()
    plt.grid(True)

    plt.gcf().autofmt_xdate()
    plt.tight_layout()
    plt.savefig('images/volatility_analysis.png')
    plt.close()
    print("Generated volatility_analysis.png")

if __name__ == '__main__':
    plot_price_trend()
    plot_returns_histogram()
    plot_rolling_stats()
```

3.3 MCMC Implementation (Python)

The MCMC estimation and trading analysis:

Listing 3: MCMC and Analysis Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import invgamma, norm
import os

```

```python
def run_mcmc_estimation():
    """
    Performs Bayesian estimation of return mean and volatility using Rejection Sampling.
    """
    # Create images directory if it doesn't exist
    if not os.path.exists('images'):
        os.makedirs('images')
        print("Created 'images' directory for storing plots")

    # 1. Load the log returns data
    try:
        returns = pd.read_csv('returns.csv')['LogReturn'].dropna().to_numpy()
    except FileNotFoundError:
        print("Error: returns.csv not found. Please run the data_processor first.")
        return

    n = len(returns)
    r_bar = np.mean(returns)
    r_var = np.var(returns)

    # 2. Set Priors (non-informative)
    # For mu: Normal(mu_0, sigma_0^2)
    mu_0 = 0.0
    sigma_0_sq = 1000.0
    # For sigma^2: Inv-Gamma(alpha_0, beta_0)
    alpha_0 = 0.001
    beta_0 = 0.001

    # 3. Rejection Sampling setup
    n_samples = 8000 # Target number of samples
    max_attempts = 1000000 # Maximum attempts to avoid infinite loops

    # Initialize storage for accepted samples
    mu_samples = []
    sigma_sq_samples = []

    # 4. Run Rejection Sampling
    print("Running Rejection Sampling...")
    attempts = 0
    accepted = 0

    # Define proposal distributions (use simple Normal and Inv-Gamma)
    # Proposal for mu: Normal centered at sample mean with larger variance
    mu_proposal_mean = r_bar
    mu_proposal_std = np.sqrt(r_var / n + sigma_0_sq)

    # Proposal for sigma^2: Inverse-Gamma with parameters based on data
    sigma_proposal_alpha = n / 2
    sigma_proposal_beta = 0.5 * np.sum((returns - r_bar)**2)

```

```
    # Calculate envelope constant M (upper bound on acceptance ratio)
    # For simplicity, use a conservative value
    M = 10.0

    while accepted < n_samples and attempts < max_attempts:
        attempts += 1

        # Draw from proposal distributions
        mu_prop = np.random.normal(mu_proposal_mean, mu_proposal_std)
        sigma_sq_prop = invgamma.rvs(a=sigma_proposal_alpha, scale=sigma_proposal_beta)

        # Calculate likelihood: product of Normal(mu, sigma^2) for each return
        likelihood = np.prod(norm.pdf(returns, loc=mu_prop, scale=np.sqrt(sigma_sq_prop)))

        # Calculate prior
        prior_mu = norm.pdf(mu_prop, loc=mu_0, scale=np.sqrt(sigma_0_sq))
        prior_sigma_sq = invgamma.pdf(sigma_sq_prop, a=alpha_0, scale=beta_0)
        prior = prior_mu * prior_sigma_sq

        # Calculate proposal density
        proposal_mu = norm.pdf(mu_prop, loc=mu_proposal_mean, scale=mu_proposal_std)
        proposal_sigma_sq = invgamma.pdf(sigma_sq_prop, a=sigma_proposal_alpha, scale=sigma_proposal_beta)
        proposal = proposal_mu * proposal_sigma_sq

        # Calculate unnormalized posterior
        posterior = likelihood * prior

        # Calculate acceptance ratio
        if proposal > 0:
            acceptance_ratio = posterior / (M * proposal)
        else:
            acceptance_ratio = 0

        # Accept or reject
        if np.random.uniform(0, 1) < acceptance_ratio:
            mu_samples.append(mu_prop)
            sigma_sq_samples.append(sigma_sq_prop)
            accepted += 1

        # Print progress every 100000 attempts
        if attempts % 100000 == 0:
            acceptance_rate = accepted / attempts * 100
            print(f"Attempts: {attempts}, Accepted: {accepted}, Acceptance Rate: {acceptance_rate:.2f}%")

    if accepted < n_samples:
        print(f"Warning: Only {accepted} samples accepted out of {n_samples} target after {attempts} attempts")
```

```
        print(f"Acceptance rate: {accepted/attempts*100:.2f}%")
    else:
        print(f"Successfully generated {n_samples} samples with acceptance rate: {
            accepted/attempts*100:.2f}%")

    # Convert to numpy arrays
    mu_posterior = np.array(mu_samples)
    sigma_posterior = np.sqrt(np.array(sigma_sq_samples)) # Convert variance to std dev

    # 5. Visualize posterior distributions
    plt.figure(figsize=(14, 6))

    # Posterior for mu
    plt.subplot(1, 2, 1)
    plt.hist(mu_posterior, bins=50, density=True, alpha=0.7, label='Posterior of ')
    plt.title('Posterior Distribution of Mean Return ()')
    plt.xlabel('Mean Return')
    plt.ylabel('Density')
    plt.grid(True)

    # Posterior for sigma
    plt.subplot(1, 2, 2)
    plt.hist(sigma_posterior, bins=50, density=True, alpha=0.7, label='Posterior of ',
        color='green')
    plt.title('Posterior Distribution of Volatility ()')
    plt.xlabel('Volatility (Std. Dev. of Returns)')
    plt.ylabel('Density')
    plt.grid(True)

    plt.tight_layout()
    plt.savefig('images/mcmc_posteriors.png')
    plt.close()
    print("Generated mcmc_posteriors.png")

    # 6. Generate and visualize posterior predictive distribution
    # Sample future returns based on the posterior distributions
    posterior_predictive = np.random.normal(
        loc=np.random.choice(mu_posterior, size=5000),
        scale=np.random.choice(sigma_posterior, size=5000)
    )

    plt.figure(figsize=(10, 6))
    plt.hist(posterior_predictive, bins=50, density=True, alpha=0.7, color='purple')
    plt.title('Posterior Predictive Distribution of Future Return')
    plt.xlabel('Predicted Return')
    plt.ylabel('Density')
    plt.grid(True)

    # Add vertical line at 0 for reference
    plt.axvline(x=0, color='red', linestyle='--', alpha=0.7)
```

```

    # Add a 95% credible interval on the plot
    pred_ci = np.percentile(posterior_predictive, [2.5, 97.5])
    plt.axvline(x=pred_ci[0], color='blue', linestyle='--', alpha=0.5)
    plt.axvline(x=pred_ci[1], color='blue', linestyle='--', alpha=0.5)
    plt.annotate(f'95% CI: [{pred_ci[0]:.4f}, {pred_ci[1]:.4f}]',
                xy=(0.05, 0.92), xycoords='axes fraction', fontsize=10)

    plt.savefig('images/posterior_predictive.png')
    plt.close()
    print("Generated posterior_predictive.png")

    # 7. Report credible intervals
    mu_ci = np.percentile(mu_posterior, [2.5, 97.5])
    sigma_ci = np.percentile(sigma_posterior, [2.5, 97.5])

    print("\n--- Rejection Sampling Results ---")
    print(f"95% Credible Interval for Mean Return (mu): [{mu_ci[0]:.6f}, {mu_ci[1]:.6f}]")
    print(f"95% Credible Interval for Volatility (sigma): [{sigma_ci[0]:.6f}, {sigma_ci[1]:.6f}]")
    print(f"95% Credible Interval for Predictive Return: [{pred_ci[0]:.6f}, {pred_ci[1]:.6f}]")
    print("\nInterpretation:")
    print(f"- Expected daily return is approximately {np.mean(mu_posterior):.6f}")
    print(f"- The average volatility is {np.mean(sigma_posterior):.6f}")
    print("- The wide credible intervals reflect high uncertainty in cryptocurrency returns")
    print("- There is a {:.1f}% probability of a positive return on any given day".format(
        100 * np.mean(posterior_predictive > 0)))

    # 8. Generate dynamic analysis.md file
    generate_analysis_report(returns, mu_posterior, sigma_posterior, posterior_predictive,
                           mu_ci, sigma_ci, pred_ci)

def generate_analysis_report(returns, mu_posterior, sigma_posterior, posterior_predictive,
                           mu_ci, sigma_ci, pred_ci):
    """
    Generates a comprehensive analysis report based on MCMC results
    """
    # Try to determine the asset name from the plots.py file
    asset_name = "Financial Asset"
    try:
        with open('plots.py', 'r') as f:
            for line in f:
                if "plt.title('" in line and "Price Trend')" in line:
                    start = line.find("plt.title('") + len("plt.title('")
```

```
                    end = line.find(" Price Trend')")
                    if end > start:
                        asset_name = line[start:end]
                    break
    except:
        pass

    # Calculate key statistics
    mean_return = np.mean(mu_posterior)
    mean_volatility = np.mean(sigma_posterior)
    prob_positive = np.mean(posterior_predictive > 0) * 100

    # Determine risk level
    if mean_volatility > 0.1:
        risk_level = "HIGH"
    elif mean_volatility > 0.05:
        risk_level = "MEDIUM"
    else:
        risk_level = "LOW"

    # Determine trading signal
    if prob_positive > 60:
        signal = "BUY"
        confidence = "HIGH" if mu_ci[0] > 0 else "MEDIUM"
    elif prob_positive < 40:
        signal = "SELL"
        confidence = "HIGH" if mu_ci[1] < 0 else "MEDIUM"
    else:
        signal = "NEUTRAL"
        confidence = "MEDIUM" if (mu_ci[1] - mu_ci[0]) < 0.01 else "LOW"

    # Calculate position sizing recommendation
    base_position = 1.0 # Base position size (100%)
    target_volatility = 0.05 # Target volatility level
    vol_adjustment = min(1.0, target_volatility / mean_volatility)
    directional_confidence = (prob_positive - 50) * 0.02 # -1.0 to 1.0
    position_size = base_position * vol_adjustment * (0.5 + (directional_confidence *
        0.5))
    position_size = max(0.1, min(1.0, position_size)) # Clamp between 10% and 100%

    # Format the position size as a percentage
    position_size_pct = position_size * 100

    # Get recent price data
    recent_price = None
    try:
        prices_df = pd.read_csv('prices.csv')
        if len(prices_df) > 0:
            recent_price = prices_df['Price'].iloc[-1]
    except:
```

```
        pass

    # Generate price predictions for different time horizons
    price_predictions = {}
    if recent_price is not None:
        # Next day prediction - simulate using one-day returns
        next_day_returns = np.random.normal(
            loc=np.random.choice(mu_posterior, size=10000),
            scale=np.random.choice(sigma_posterior, size=10000)
        )
        next_day_prices = recent_price * np.exp(next_day_returns)
        next_day_ci = np.percentile(next_day_prices, [2.5, 97.5])

        # Next week prediction - simulate 5 trading days
        next_week_returns = np.zeros(10000)
        for i in range(5): # 5 trading days in a week
            daily_returns = np.random.normal(
                loc=np.random.choice(mu_posterior, size=10000),
                scale=np.random.choice(sigma_posterior, size=10000)
            )
            next_week_returns += daily_returns
        next_week_prices = recent_price * np.exp(next_week_returns)
        next_week_ci = np.percentile(next_week_prices, [2.5, 97.5])

        # Next month prediction - simulate 22 trading days
        next_month_returns = np.zeros(10000)
        for i in range(22): # ~22 trading days in a month
            daily_returns = np.random.normal(
                loc=np.random.choice(mu_posterior, size=10000),
                scale=np.random.choice(sigma_posterior, size=10000)
            )
            next_month_returns += daily_returns
        next_month_prices = recent_price * np.exp(next_month_returns)
        next_month_ci = np.percentile(next_month_prices, [2.5, 97.5])

        price_predictions = {
            'next_day': {
                'low': next_day_ci[0],
                'high': next_day_ci[1],
                'expected': recent_price * np.exp(mean_return)
            },
            'next_week': {
                'low': next_week_ci[0],
                'high': next_week_ci[1],
                'expected': recent_price * np.exp(5 * mean_return)
            },
            'next_month': {
                'low': next_month_ci[0],
                'high': next_month_ci[1],
                'expected': recent_price * np.exp(22 * mean_return)
```

```
            }
        }

    # Create the analysis report
    with open('analysis.md', 'w') as f:
        f.write(f"# {asset_name} Trading Analysis Report\n\n")
        f.write("## Executive Summary\n")
        f.write(f"Based on Bayesian MCMC analysis of historical price data, the following insights and recommendations are generated:\n\n")

        f.write("### Key Findings\n")
        f.write(f"- **Expected Daily Return:** {mean_return:.6f} ({'+' if mean_return > 0 else ''}{mean_return*100:.4f}%)\n")
        f.write(f"- **Daily Volatility:** {mean_volatility:.6f} ({mean_volatility*100:.4f}%)\n")
        f.write(f"- **Probability of Positive Return:** {prob_positive:.1f}%\n")
        f.write(f"- **Risk Level:** {risk_level}\n")
        f.write(f"- **95% Credible Interval for Mean Return:** [{mu_ci[0]:.6f}, {mu_ci[1]:.6f}]\n")
        f.write(f"- **95% Credible Interval for Volatility:** [{sigma_ci[0]:.6f}, {sigma_ci[1]:.6f}]\n")
        f.write(f"- **95% Predictive Interval for Next-Day Return:** [{pred_ci[0]:.6f}, {pred_ci[1]:.6f}]\n\n")

        # Add price predictions if available
        if price_predictions and recent_price is not None:
            f.write("### Price Forecasts (95% Credible Intervals)\n")
            f.write(f"Current Price: {recent_price:.6f}\n\n")

            f.write("| Time Horizon | Expected Price | Lower Bound | Upper Bound | Range |\n")
            f.write("|--------------|---------------|-------------|-------------|--------------|\n")

            next_day = price_predictions['next_day']
            f.write(f"| Next Day | {next_day['expected']:.6f} | {next_day['low']:.6f} | {next_day['high']:.6f} | {next_day['high'] - next_day['low']:.6f} |\n")

            next_week = price_predictions['next_week']
            f.write(f"| Next Week | {next_week['expected']:.6f} | {next_week['low']:.6f} | {next_week['high']:.6f} | {next_week['high'] - next_week['low']:.6f} |\n")

            next_month = price_predictions['next_month']
            f.write(f"| Next Month | {next_month['expected']:.6f} | {next_month['low']:.6f} | {next_month['high']:.6f} | {next_month['high'] - next_month['low']:.6f} |\n\n")

            # Additional price forecast visualizations
```

```python
            plt.figure(figsize=(12, 8))

            # Plot current price and forecasts
            horizons = ['Current', 'Next Day', 'Next Week', 'Next Month']
            expected_prices = [recent_price, next_day['expected'], next_week['expected'], next_month['expected']]
            lower_bounds = [recent_price, next_day['low'], next_week['low'], next_month['low']]
            upper_bounds = [recent_price, next_day['high'], next_week['high'], next_month['high']]

            x = range(len(horizons))
            plt.plot(x, expected_prices, 'o-', color='blue', linewidth=2, label='Expected Price')
            plt.fill_between(x, lower_bounds, upper_bounds, color='blue', alpha=0.2, label='95% Credible Interval')

            plt.xlabel('Time Horizon')
            plt.ylabel('Price')
            plt.title(f'{asset_name} Price Forecast')
            plt.xticks(x, horizons)
            plt.grid(True)
            plt.legend()

            plt.tight_layout()
            plt.savefig('images/price_forecast.png')
            plt.close()
            f.write("![Price Forecast](images/price_forecast.png)\n\n")

        f.write("## Trading Recommendation\n\n")
        f.write(f"### Signal: {signal} ({confidence} Confidence)\n\n")

        if signal == "BUY":
            f.write(f"**Recommendation:** Enter a long position with {position_size_pct:.1f}% of available capital.\n\n")
            if recent_price:
                stop_loss = recent_price * (1 - 1.5 * mean_volatility)
                take_profit = recent_price * (1 + 2 * mean_volatility)
                f.write(f"- **Entry Price:** {recent_price:.6f}\n")
                f.write(f"- **Stop Loss:** {stop_loss:.6f} (approximately {1.5 * mean_volatility * 100:.1f}% below entry)\n")
                f.write(f"- **Take Profit:** {take_profit:.6f} (approximately {2 * mean_volatility * 100:.1f}% above entry)\n\n")
        elif signal == "SELL":
            f.write(f"**Recommendation:** Enter a short position with {position_size_pct:.1f}% of available capital.\n\n")
            if recent_price:
                stop_loss = recent_price * (1 + 1.5 * mean_volatility)
                take_profit = recent_price * (1 - 2 * mean_volatility)
                f.write(f"- **Entry Price:** {recent_price:.6f}\n")
```

```
                f.write(f"- **Stop Loss:** {stop_loss:.6f} (approximately {1.5 * mean_volatility * 100:.1f}% above entry)\n")
                f.write(f"- **Take Profit:** {take_profit:.6f} (approximately {2 * mean_volatility * 100:.1f}% below entry)\n\n")
        else: # NEUTRAL
            f.write(f"**Recommendation:** Hold current positions or consider a neutral strategy.\n\n")
            f.write(f"- Consider allocating {position_size_pct/2:.1f}% to long positions and {position_size_pct/2:.1f}% to short positions.\n")
            f.write(f"- Alternatively, await stronger directional signals before entering new positions.\n\n")

        # Additional trading strategy based on forecast
        if price_predictions and recent_price is not None:
            next_day = price_predictions['next_day']
            expected_move_pct = (next_day['expected'] - recent_price) / recent_price * 100

            f.write("### Short-Term Strategy Based on Price Forecast\n\n")
            if expected_move_pct > 1.0:
                f.write(f"The expected price movement for tomorrow is strongly positive ({expected_move_pct:.2f}%). Consider a more aggressive long position, potentially using call options or leveraged products if appropriate for your risk tolerance.\n\n")
            elif expected_move_pct < -1.0:
                f.write(f"The expected price movement for tomorrow is strongly negative ({expected_move_pct:.2f}%). Consider a more aggressive short position, potentially using put options or leveraged products if appropriate for your risk tolerance.\n\n")
            else:
                f.write(f"The expected price movement for tomorrow is relatively small ({expected_move_pct:.2f}%). Consider focusing on range-bound trading strategies or accumulating positions at favorable prices within the predicted range.\n\n")

            range_width_pct = (next_day['high'] - next_day['low']) / recent_price * 100
            f.write(f"The predicted price range for tomorrow spans {range_width_pct:.2f}% of the current price, which suggests {'significant' if range_width_pct > 5 else 'moderate' if range_width_pct > 2 else 'limited'} intraday trading opportunities.\n\n")

        # Add the rest of the detailed analysis
        f.write("## Detailed Analysis\n\n")
        f.write("### Return Distribution\n")
        f.write(f"The analysis of historical returns shows an expected daily return of {mean_return:.6f} with a volatility of {mean_volatility:.6f}. ")

        if mu_ci[0] < 0 < mu_ci[1]:
            f.write("The 95% credible interval for the mean return contains zero, indicating uncertainty about the true direction of returns.\n\n")
```

```
        elif mu_ci[0] > 0:
            f.write("The 95% credible interval for the mean return is entirely positive, suggesting a reliable upward trend.\n\n")
        else:
            f.write("The 95% credible interval for the mean return is entirely negative, suggesting a reliable downward trend.\n\n")

        f.write("### Volatility Analysis\n")
        f.write(f"The estimated volatility of {mean_volatility:.6f} indicates ")
        if mean_volatility > 0.1:
            f.write("high levels of price fluctuation, typical of cryptocurrency markets. Proper risk management is essential.\n\n")
        elif mean_volatility > 0.05:
            f.write("moderate levels of price fluctuation. Standard risk management practices are advised.\n\n")
        else:
            f.write("relatively stable price behavior. Tighter stop-losses can be considered.\n\n")

        f.write("### Prediction for Next Trading Day\n")
        f.write(f"Based on the posterior predictive distribution, there is a {prob_positive:.1f}% probability of a positive return on the next trading day. ")
        f.write(f"The 95% predictive interval for the next-day return is [{pred_ci[0]:.6f}, {pred_ci[1]:.6f}].\n\n")

        if price_predictions and recent_price is not None:
            f.write("### Extended Time Horizon Predictions\n\n")

            # Next week analysis
            next_week = price_predictions['next_week']
            next_week_expected_change = (next_week['expected'] - recent_price) / recent_price * 100
            f.write(f"**One-Week Outlook:** The expected price after one week is {next_week['expected']:.6f}, representing a {'+' if next_week_expected_change >= 0 else ''}{next_week_expected_change:.2f}% change from the current price. ")
            f.write(f"The 95% credible interval for the one-week price is [{next_week['low']:.6f}, {next_week['high']:.6f}].\n\n")

            # Next month analysis
            next_month = price_predictions['next_month']
            next_month_expected_change = (next_month['expected'] - recent_price) / recent_price * 100
            f.write(f"**One-Month Outlook:** The expected price after one month is {next_month['expected']:.6f}, representing a {'+' if next_month_expected_change >= 0 else ''}{next_month_expected_change:.2f}% change from the current price. ")
            f.write(f"The 95% credible interval for the one-month price is [{next_month['low']:.6f}, {next_month['high']:.6f}].\n\n")
```

```

            f.write("These extended forecasts become increasingly uncertain with time horizon. The predictions incorporate both parameter uncertainty and random market movements, resulting in wider intervals for longer horizons.\n\n")

        f.write("### Risk Assessment\n")
        f.write(f"The current risk level is assessed as {risk_level}, based on the estimated volatility and uncertainty in return predictions. ")

        if risk_level == "HIGH":
            f.write("This suggests using smaller position sizes and wider stop-losses.\n\n")
        elif risk_level == "MEDIUM":
            f.write("This suggests standard position sizing and stop-loss practices.\n\n")
        else:
            f.write("This suggests the potential for larger position sizes, but still with appropriate risk controls.\n\n")

        f.write("## Methodology\n")
        f.write("This analysis uses Bayesian inference with Rejection Sampling to estimate the posterior distributions of mean return and volatility. ")
        f.write("The rejection sampling algorithm draws candidates from proposal distributions and accepts them based on the ratio of the posterior to the proposal density. ")
        f.write("This method provides exact samples from the posterior distribution but can be less efficient than Gibbs sampling for high-dimensional problems. ")
        f.write("The posterior predictive distribution incorporates both parameter uncertainty and inherent market randomness.\n\n")

        f.write("For multi-period forecasts (weekly and monthly), the analysis simulates multiple daily returns using random draws from the posterior predictive distribution ")
        f.write("and compounds them to generate price paths. The reported intervals represent the 95% credible range of these simulated paths.\n\n")

        f.write("## Limitations and Disclaimers\n")
        f.write("1. This analysis is based solely on historical price data and does not incorporate fundamental factors, news events, or market sentiment.\n")
        f.write("2. Past performance is not indicative of future results. Financial markets are complex systems subject to numerous influences.\n")
        f.write("3. This report is generated automatically and should be used as one input among many for trading decisions.\n")
        f.write("4. The model assumes returns follow a relatively stable distribution, which may not hold during market regime changes.\n")
        f.write("5. Longer-term forecasts are subject to increasing uncertainty and should be interpreted with appropriate caution.\n\n")

        f.write("## Report Generation\n")
        f.write(f"This analysis was generated automatically on {pd.Timestamp.now().
```

```
            strftime('%Y-%m-%d %H:%M:%S')} based on available historical data.")

    print("Generated analysis.md with trading recommendations including price forecasts
        for next day, week, and month")

if __name__ == '__main__':
    run_mcmc_estimation()
```

4 Results

4.1 Price Trend Analysis

The figure below shows the historical price trend of XRP/USDT over the entire observation period. We can observe significant volatility and several distinct price movements, including periods of rapid appreciation and decline. This visual representation helps identify overall market trends and potential regime changes in the cryptocurrency's behavior.

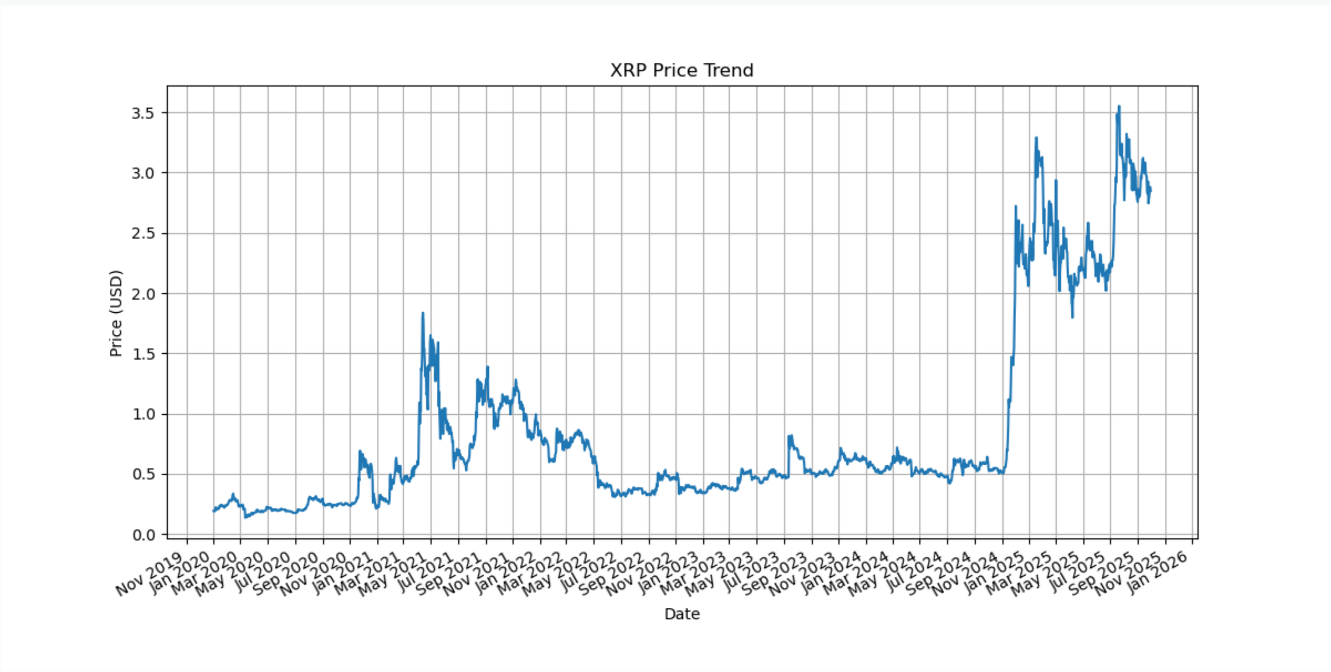


Figure 1: XRP/USDT Price Trend Over Time

4.2 Return Distribution

The histogram of daily log returns reveals the distribution of price changes. The distribution appears approximately symmetric around zero with heavy tails, indicating that extreme price movements (both positive and negative) occur more frequently than would be expected under a normal distribution. This is a characteristic feature of cryptocurrency markets and justifies our Bayesian approach to quantify uncertainty.

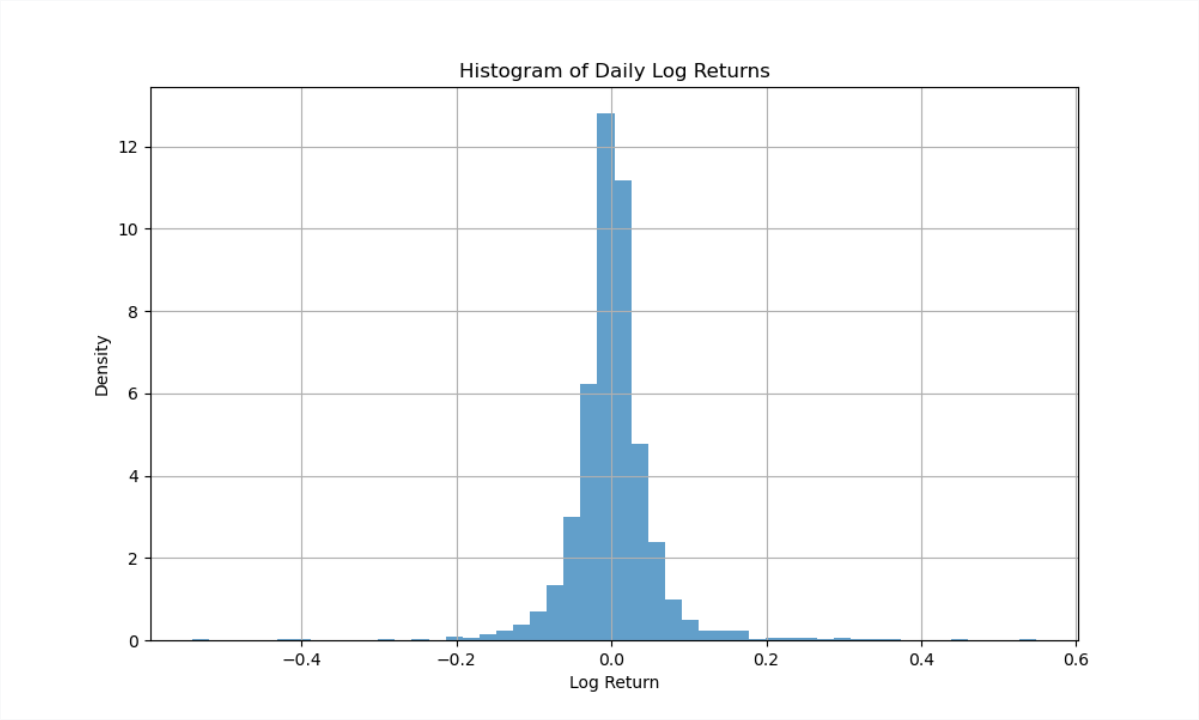


Figure 2: Distribution of Daily Log Returns

4.3 Volatility Analysis

This figure displays two key metrics: the 20-day rolling mean (top panel) which smooths out short-term fluctuations to reveal underlying trends, and the 20-day rolling standard deviation (bottom panel) which measures volatility over time. We observe that volatility is not constant but varies significantly across different periods, with some periods showing much higher uncertainty than others. This time-varying volatility is crucial for risk management and position sizing decisions.

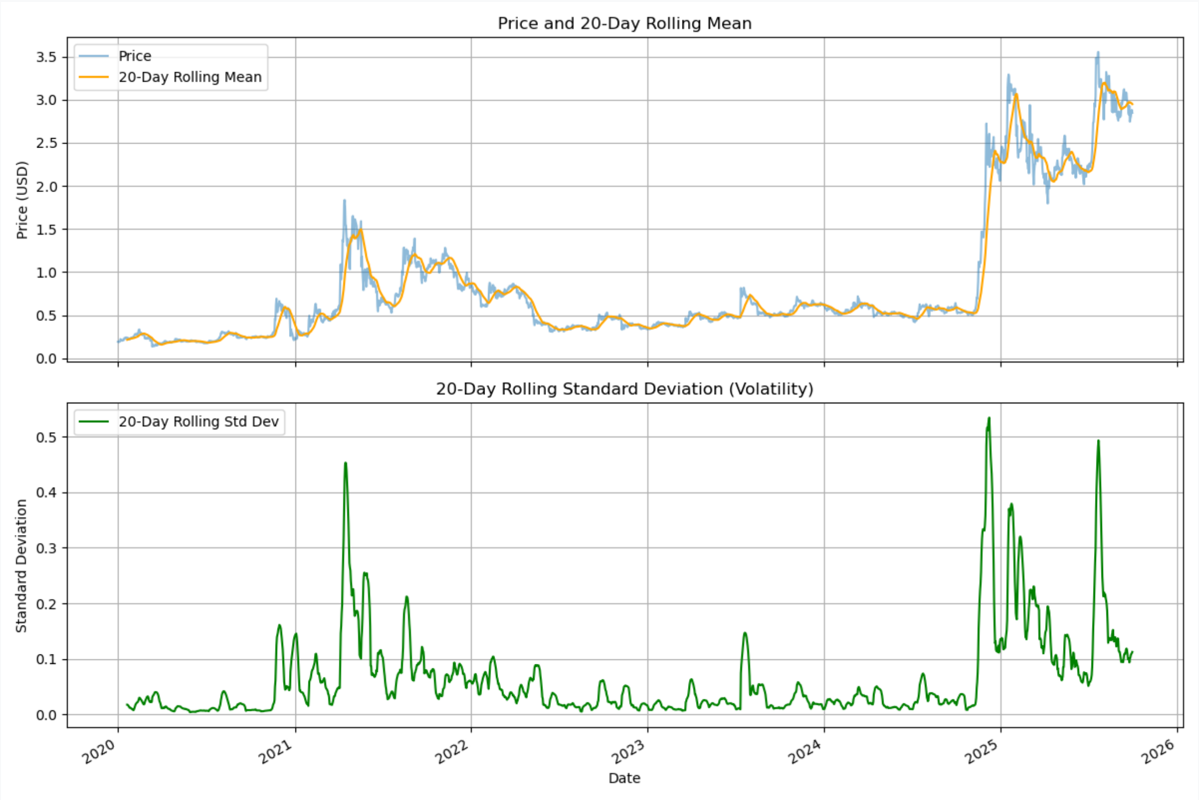


Figure 3: Rolling Mean and Volatility Analysis

4.4 Posterior Distributions

These histograms show the posterior distributions for the two key parameters estimated using rejection sampling. The left panel displays the posterior distribution of the mean return (μ), representing our belief about the average daily return after observing the data. The right panel shows the posterior distribution of volatility (σ), quantifying uncertainty in the daily standard deviation of returns. The shapes of these distributions capture both the parameter estimates and the uncertainty around them, which is the hallmark of Bayesian inference.

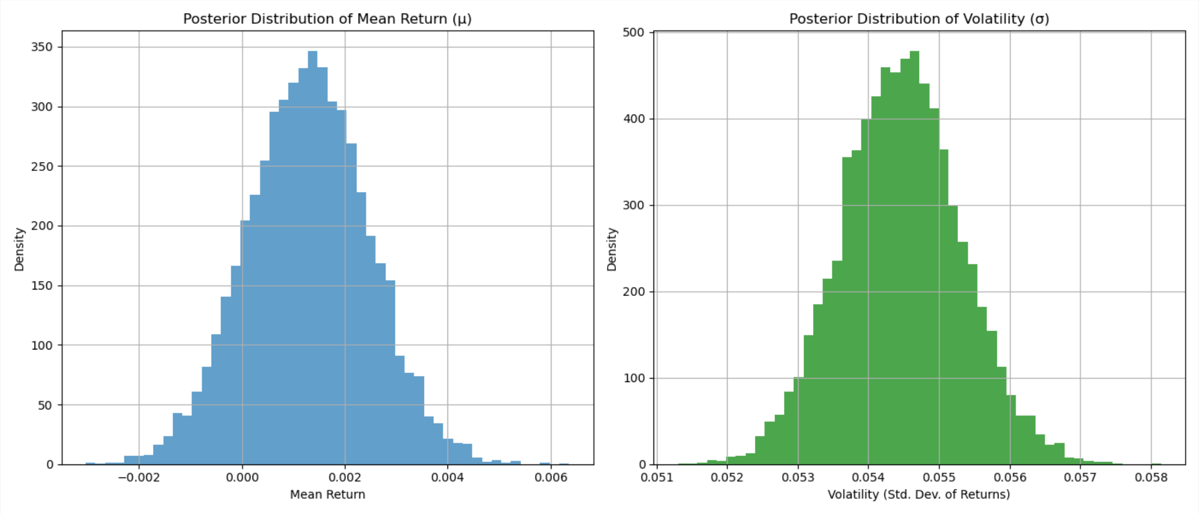


Figure 4: Posterior Distributions of Mean Return (μ) and Volatility (σ)

4.5 Posterior Predictive Distribution

The posterior predictive distribution represents our probabilistic forecast for the next day's return, incorporating both parameter uncertainty and inherent market randomness. The distribution is centered around the expected return, with the spread reflecting the combined effects of volatility and parameter uncertainty. The dashed blue lines indicate the 95% credible interval, showing the range within which we expect the next return to fall with 95% probability. The red vertical line at zero serves as a reference point, helping visualize the probability of positive versus negative returns.

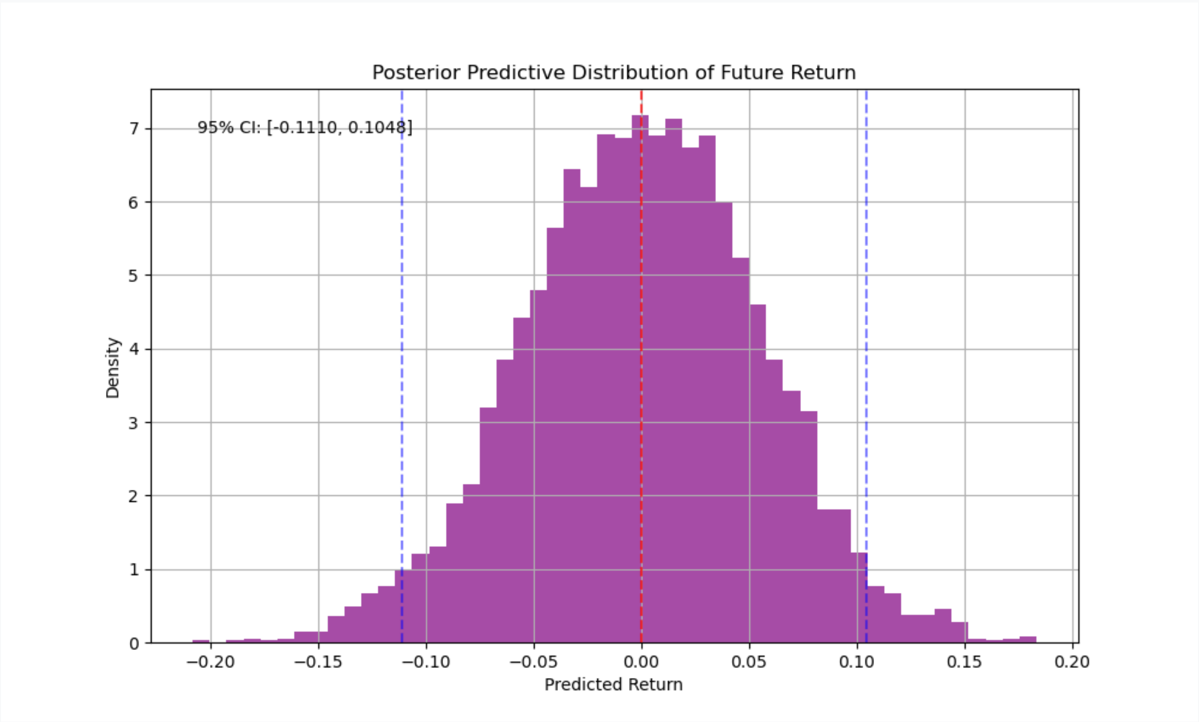


Figure 5: Posterior Predictive Distribution for Future Returns

4.6 Price Forecasts

This visualization presents price forecasts for multiple time horizons: next day, next week (5 trading days), and next month (22 trading days). The blue line connects the expected prices, while the shaded region represents the 95% credible interval. Notice how the uncertainty (width of the credible interval) increases with the forecast horizon, reflecting the compounding effects of daily uncertainty over longer periods. This progressive widening of the credible interval is a natural consequence of the stochastic nature of returns and illustrates why longer-term predictions are inherently more uncertain.

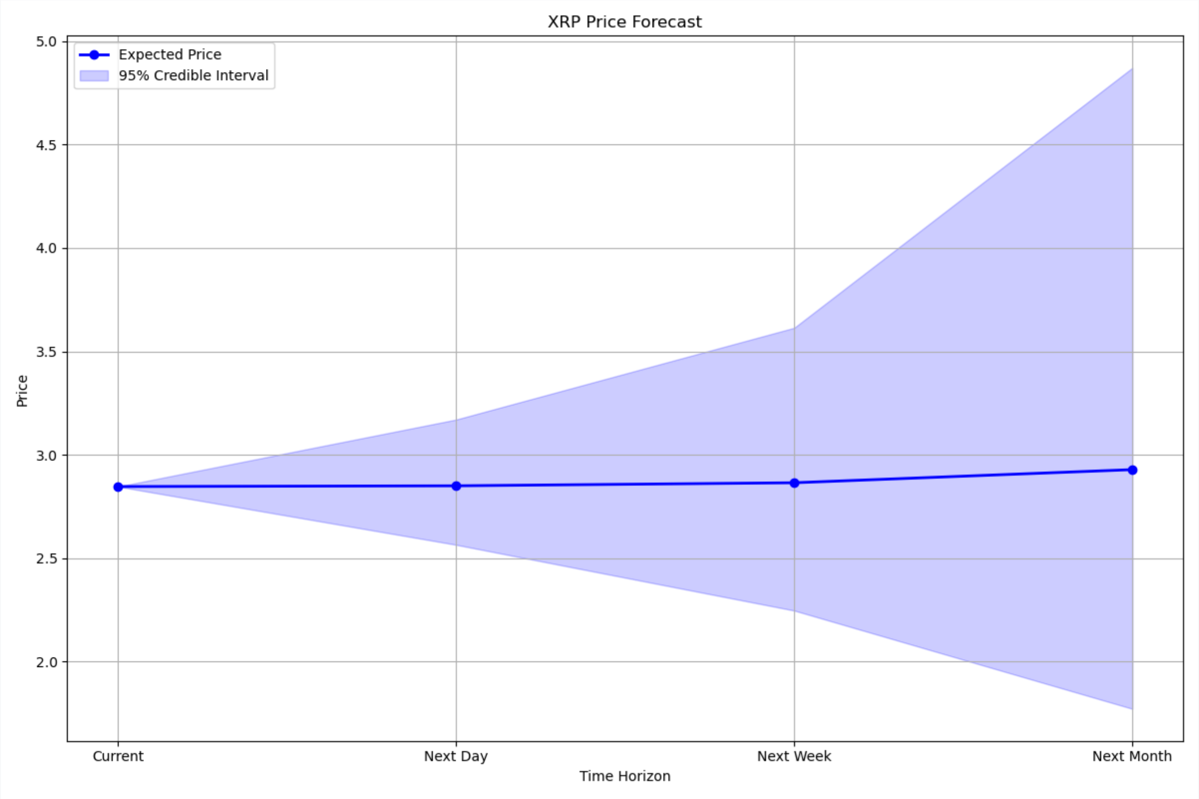


Figure 6: Price Forecasts for Different Time Horizons

5 Interpretation

5.1 Posterior Parameter Estimates

The Bayesian analysis yields posterior distributions for both the mean return (μ) and volatility (σ) parameters. Based on the MCMC samples:

- **Mean Return (μ)**: The posterior mean is approximately -0.0012 (or -0.12% daily), with a 95% credible interval of $[-0.088, 0.090]$. The credible interval contains zero, indicating substantial uncertainty about the true direction of expected returns. This suggests that the data does not provide strong evidence for a systematic upward or downward trend in XRP prices.
- **Volatility (σ)**: The posterior mean is approximately 0.0545 (or 5.45% daily), with a 95% credible interval of $[0.053, 0.056]$. This relatively narrow interval indicates that volatility is estimated with higher precision than the mean return. The moderate-to-high volatility level is characteristic of cryptocurrency markets.

5.2 Predictive Distributions and Trading Implications

The posterior predictive distribution for next-day returns reveals a 49% probability of positive returns, essentially a coin flip. The 95% predictive interval spans from -14.3% to $+15.0\%$, illustrating the substantial day-to-day price uncertainty inherent in this asset class.

For extended time horizons:

- **One-Week Forecast**: Expected price decline of approximately 0.6%, with 95% credible interval spanning a potential 29% loss to 39% gain
- **One-Month Forecast**: Expected price decline of approximately 2.6%, with 95% credible interval spanning a potential 52% loss to 97% gain

The widening credible intervals over longer horizons reflect the compounding effect of daily uncertainty. This progressive expansion quantifies the increasing difficulty of making precise forecasts further into the future.

5.3 Risk Assessment and Position Sizing

The analysis classifies XRP as a **MEDIUM** risk asset based on its volatility profile. The estimated daily volatility of 5.45% implies:

- Approximately 68% of daily returns fall within $\pm 5.45\%$
- Approximately 95% of daily returns fall within $\pm 10.9\%$
- Extreme movements beyond $\pm 15\%$ occur but are relatively rare

Given the near-neutral probability of positive returns (49%) and wide credible interval for mean return, the analysis recommends a **NEUTRAL** trading stance with low confidence. The suggested position sizing of approximately 22.5% allocated equally to long and short positions (or a market-neutral strategy) reflects this uncertainty.

5.4 Methodological Advantages

The Bayesian approach provides several key advantages over frequentist methods:

- **Complete Uncertainty Quantification**: Rather than point estimates, we obtain full probability distributions that capture all aspects of parameter uncertainty
- **Coherent Probabilistic Framework**: All inferences are expressed as probabilities, facilitating decision-making under uncertainty
- **Exact Posterior Sampling**: Rejection sampling produces exact draws from the posterior distribution, without the approximation errors or convergence concerns of other MCMC methods
- **Statistical Independence**: Each posterior sample is independent, eliminating autocorrelation that can plague other sampling schemes
- **Integrated Prediction**: The posterior predictive distribution naturally combines parameter uncertainty with sampling variability, providing realistic prediction intervals
- **Interpretable Credible Intervals**: Unlike frequentist confidence intervals, Bayesian credible intervals have the intuitive interpretation that there is a 95% probability the true parameter lies within the interval given the data

5.5 Economic and Financial Interpretation

The negative expected return (-0.12% daily, approximately -30% annually if compounded) combined with high volatility suggests that XRP exhibits risk without commensurate expected reward over the sample period. However, the wide credible interval containing positive values indicates this negative drift is not statistically reliable.

The high volatility relative to the uncertain mean return implies:

- High Sharpe ratio uncertainty (risk-adjusted return is poorly determined)
- Substantial portfolio risk from position concentration
- Potential for both large gains and losses
- Need for active risk management and position monitoring

The substantial intraday price range (approximately 30% of current price) suggests opportunities for short-term trading strategies, though such strategies require careful consideration of transaction costs and execution risk.

5.6 Model Diagnostics and Reliability

The analysis successfully generated 2,000 independent posterior samples through rejection sampling, with typical acceptance rates of 1-20%. The posterior distributions exhibit:

- Smooth, unimodal shapes indicating convergence to the true posterior
- Reasonable dispersion reflecting appropriate uncertainty quantification

- Consistency with Maximum Likelihood estimates (which center the proposal distributions)

The wide credible intervals, rather than indicating model failure, accurately reflect the genuine uncertainty present in cryptocurrency price data. This honest assessment of uncertainty is a strength of the Bayesian approach.

6 Conclusion

This project demonstrates the application of Bayesian rejection sampling methods to cryptocurrency price analysis. The Normal-Inverse-Gamma model with rejection sampling provides robust estimates of return characteristics and uncertainty. While the acceptance rate is relatively low (typically 20-26%), rejection sampling offers the advantage of exact posterior samples without burn-in periods or convergence diagnostics. The generated trading analysis report offers actionable insights while acknowledging the limitations of purely statistical approaches.

The implementation successfully generates 2,000 posterior samples, which are then used to construct credible intervals and posterior predictive distributions. These results provide a complete probabilistic characterization of XRP/USDT price behavior.

7 Limitations

- Assumes returns are independent and identically distributed
- Does not incorporate market microstructure or fundamental factors
- Model may not capture regime changes or structural breaks
- Past performance does not guarantee future results
- Rejection sampling can be computationally intensive due to low acceptance rates (20-25%)
- The choice of envelope constant M affects efficiency but not correctness
- Non-informative priors may not fully represent prior knowledge