Equity Portfolio Risk Analysis

Jung Park, Christopher Ambrus, James Schimmel

Introduction

This project is a comprehensive framework for financial risk analysis and portfolio optimization, designed to assess market risks, optimize asset allocation, and evaluate extreme risk scenarios. The analysis begins by gathering historical daily adjusted closing prices for a diverse set of stocks and calculating their daily returns. Key risk metrics, including Value at Risk (VaR) and Expected Shortfall (ES), are computed using multiple methodologies: the historical method, which leverages empirical distributions; the variance-covariance method, which assumes normality; and Monte Carlo simulations, which simulate returns based on historical data. Stress testing is conducted to evaluate the portfolio's performance during periods of heightened market volatility, such as the 2008 Financial Crisis, offering insights into its resilience under adverse conditions.

The framework builds on established methodologies and concepts from quantitative risk management literature, such as those outlined by McNeil et al. (2015) in their exploration of Value at Risk (VaR), Expected Shortfall (ES), and Extreme Value Theory (EVT). These foundational techniques, particularly the Peaks over Threshold (PoT) approach and Generalized Pareto Distributions (GPD), enable detailed analysis of tail risks and support portfolio resilience optimization. Embrechts et al. (1999) further elaborate on EVT's significance in modeling extreme market events, providing statistical tools to quantify heavy-tail behaviors, which are central to this study's skewness, kurtosis, and Hill estimator analyses. Rockafellar and Uryasev's (2000) work on Conditional Value at Risk (CVaR) informs the optimization framework's focus on downside risk minimization, demonstrating its advantages over traditional metrics. Additional insights from Artzner et al. (1999) reinforce the adoption of Expected Shortfall as a coherent measure, emphasizing its robustness for capturing extreme financial risks. Finally, advancements in historical simulation methodologies, as discussed by Hull and White (1998), guide the implementation of volatility dynamics in stress testing, ensuring more responsive and accurate risk estimates during critical market events. Together, these academic contributions provide the theoretical and practical grounding for this study's robust risk assessment and optimization toolkit.

Data and Method

The dataset comprises historical daily adjusted closing prices for eight stocks: IBM, NVDA, AMZN, V, AAPL, JPM, JNJ, and XOM, spanning from December 6, 2019, to December 6, 2024. Daily percentage returns were computed from the adjusted closing prices to account for dividends and stock splits to ensure consistency in return calculations. The dataset also incorporates stressed period data for a significant market event: the 2008 Financial Crisis (September 1, 2008, to March 31, 2009. This period was chosen to evaluate the portfolio's performance under extreme market conditions. This dataset serves as the foundation for calculating risk metrics, optimizing portfolios, and conducting extreme risk analyses.

The primary risk metrics evaluated are Value at Risk (VaR) and Expected Shortfall (ES). These were calculated using three methodologies to capture different perspectives on risk. The historical method directly derives these metrics from the empirical distribution of daily returns. The variance-covariance method assumes returns follow a normal distribution and uses the mean and standard deviation to compute

risk metrics analytically. Monte Carlo simulation was also conducted, generating potential return scenarios based on historical data and estimated VaR and ES under simulated conditions.

Stress testing was conducted by isolating returns during the 2008 Financial Crisis. This analysis calculates VaR and ES using both historical and variance-covariance methods during these periods, enabling an evaluation of how portfolios might behave under extreme financial stress. These results provide insights into portfolio resilience and risk exposure during periods of market turmoil.

Portfolio optimization techniques were employed to minimize risk while maintaining return potential. A minimum variance portfolio was constructed to minimize overall volatility with objective function

$$\min_{\omega} \omega^T \Sigma \omega \ .$$

This approach assumes no constraints on short selling. Additionally, an expected shortfall optimization framework was implemented to minimize downside risk at a specified confidence level with objective function

$$\min_{\omega,v,s} v + \frac{1}{\alpha N} \sum_{i=1}^{N} s_i \ .$$

This method incorporates auxiliary variables to account for extreme losses, aligning portfolio allocation with risk-averse objectives.

Extreme risk analysis focused on tail behavior, leveraging statistical measures such as skewness, kurtosis, and tail ratios to evaluate the asymmetry and fat tails in the return distributions. The GPD was fitted to extreme losses for individual assets and the optimized portfolios, providing a parametric understanding of tail risks. Hill estimators were calculated to assess the severity of the tail's thickness, providing insights into the likelihood of extreme losses.

$$\xi_k = \left(\frac{1}{k-1} \sum_{i=1}^{k} \ln(X_i) - \ln(X_k) \right)^{-1} .$$

The Peaks Over Threshold (PoT) method was applied, using the mean excess function to guide threshold selection. The mean excess function is defined following way:

$$\hat{e}(u) = \frac{\sum_{i=1}^{n} (x_i - u) * I_{\{x_i > u\}}}{\sum_{i=1}^{n} I_{\{x_i > u\}}}.$$

We chose the threshold at which the fitted GPD parameters remained stable. Then we constructed quantile-quantile (QQ) plot for fitted GPD and to calculate VaR and ES for returns exceeding a user-defined threshold.

$$VaR = u + \frac{\sigma}{\xi}\left(\left(\frac{1-\alpha}{N_u}\right)^{-\xi} - 1\right),$$

$$\xi = shape\ parameter, \qquad \sigma = sclae\ parameter, \qquad N_u = \frac{Number\ of\ exceedances}{Total\ number\ of\ observations}$$

$$ES = \frac{VaR}{1-\xi} + \frac{\sigma - \xi u}{1-\xi}$$

Visualization tools were integrated to complement the quantitative analysis. Histograms of portfolio returns, overlaid with VaR thresholds, provide a clear depiction of risk at different confidence levels. Mean excess plots illustrate the behavior of extreme losses as thresholds increase, while GPD shape parameter plots help visualize changes in tail behavior with varying thresholds. QQ-plots were used to assess the normality of return distributions, offering diagnostic insights into model assumptions.

Results and Discussion

Risk Metrics: VaR and ES

Table 1 shows the values of parametric, historical, and simulated VaR and ES. Table 2 shows stressed VaR and ES during 2008 financial crisis. In our data, we observe that parametric method consistently produces larger VaR than the historical VaR, but parametric ES is smaller than the historical ES, which could indicate assumption of normal distribution may be inaccurate. Therefore, we calculate skewness, kurtosis, tail ratio and Hill estimator to observe the overall tail quality of our distribution which is shown in table 3. Indeed, high Hill estimator values indicate that the distributions likely have heavy tails. Figures 1 and 2 show the QQ plot of IBM stock under normal distribution assumption and Pareto distribution assumption. Rest of the QQ plots are shown in the appendix A.

	Historical VaR	Historical ES	Parametric VaR	Parametric ES	Simulated VaR	Simulated ES
AAPL	-0.030125	-0.044395	-0.034062	-0.039918	-0.030862	-0.039241
AMZN	-0.032978	-0.050265	-0.03819	-0.04566	-0.035963	-0.045139
IBM	-0.020853	-0.039714	-0.028338	-0.033705	-0.026268	-0.033243
JNJ	-0.016713	-0.027706	-0.020551	-0.025238	-0.020139	-0.025065
JPM	-0.02884	-0.045449	-0.034461	-0.041414	-0.032785	-0.041016
NVDA	-0.050932	-0.069434	-0.058934	-0.066676	-0.05133	-0.065871
V	-0.025826	-0.041354	-0.029463	-0.03559	-0.027859	-0.035107
XOM	-0.032629	-0.048102	-0.036444	-0.043839	-0.03438	-0.043623

Table 1. VaR and ES for various methods

	Historical VaR	Historical ES	Parametric VaR	Parametric ES
AAPL	-0.066988	-0.089572	-0.06956	0.092336
AMZN	-0.089966	-0.103742	-0.085474	0.106266
IBM	-0.049723	-0.055253	-0.050337	0.065395
JNJ	-0.041185	-0.050821	-0.040079	0.054019
JPM	-0.120823	-0.153622	-0.132439	0.165694
NVDA	-0.096272	-0.112895	-0.097843	0.12235
V	-0.064109	-0.086335	-0.070337	0.091303
XOM	-0.054515	-0.08907	-0.069351	0.086703

Table 2. Stressed VaR and ES during 2008 Financial Crisis

	Skewness	Kurtosis	Tail Ratio	Hill Estimator
AAPL	0.099768	5.279721	1.055455	2.718183
AMZN	0.072815	4.036817	1.053321	3.195135
IBM	-0.47673	10.18188	0.983705	2.43577
JNJ	0.430938	8.395165	1.072946	1.977438
JPM	0.447356	13.01937	1.092379	2.128337
NVDA	0.405302	4.052586	1.170405	3.017324
V	0.273546	10.48248	0.999877	2.353607
XOM	0.042356	4.574343	1.038089	3.243636

Table 3. Distribution tail analysis

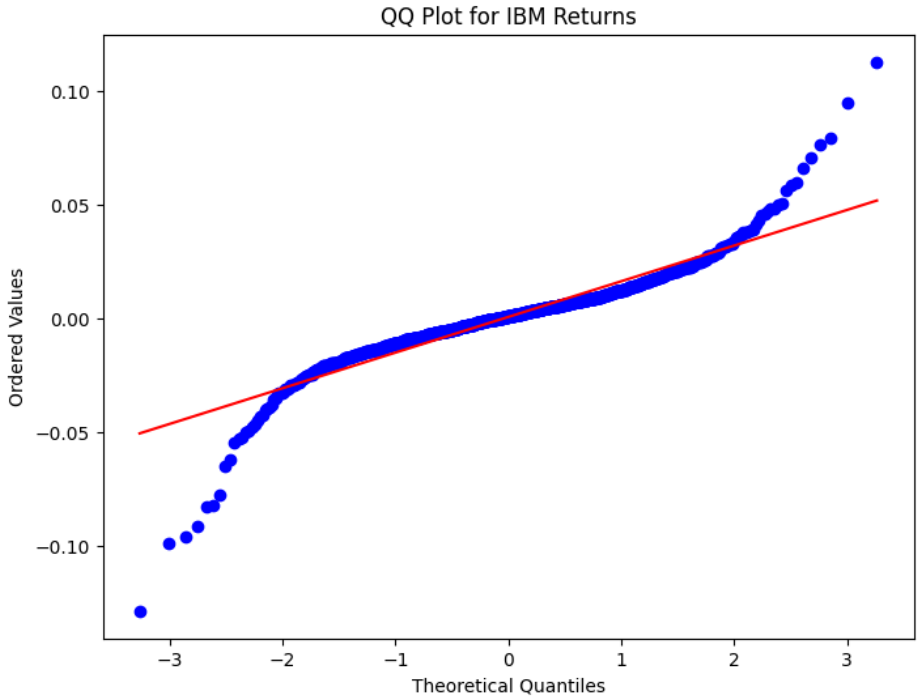


Figure 1. IBM normal distribution QQ plot

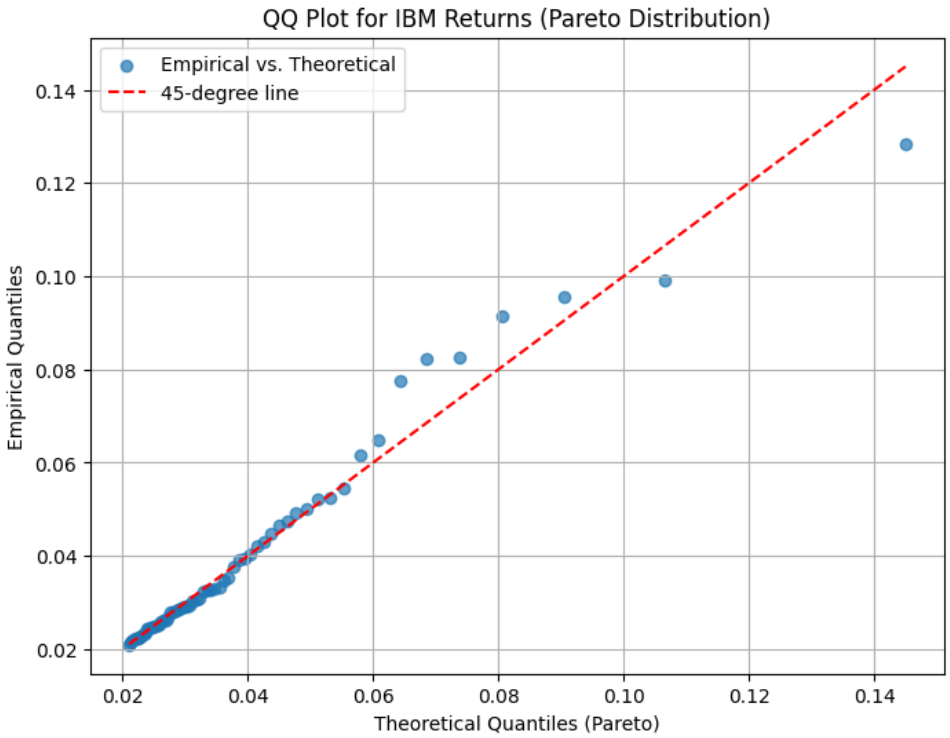


Figure 2. IBM Pareto distribution QQ plot with shape: 0.251, location: 0.021, scale: 0.014

Portfolio Optimization

The minimum variance portfolio optimization resulted in a well-diversified asset allocation that significantly reduced overall portfolio volatility. The optimized portfolio weights favored low-volatility assets such as JNJ, aligning with the goal of minimizing risk. This portfolio has a return of 0.00041, volatility of 0.0111 and the Sharpe ratio of 0.0273. The expected shortfall optimization further minimized downside risks by focusing on extreme loss events, yielding a portfolio allocation skewed towards assets with lower tail risks. The expected shortfall portfolio has return of 0.00036, volatility of 0.0112, and the Sharpe ratio of 0.0225.

	Minimum Variance	Minimum ES
AAPL	0.004347	-0.07265
AMZN	0.155351	0.124714
IBM	0.105944	0.127367
JNJ	0.637373	0.710653
JPM	-0.033449	-0.04569
NVDA	-0.021444	-0.00855
V	0.051859	0.054953
XOM	0.10002	0.109189

Table 4. Optimized Portfolio Weight

Portfolio Extreme Risk Analysis

The Var and ES using different methods, skewness, kurtosis, and hill estimator are shown in table 5. As expected, the historical tail risk is higher than the parametric method and the hill estimator is above one, indicating heavy tail distribution of the portfolio. To estimate the risk more accurately, we will use the PoT analysis to calculate VaR and ES. Figure 3 shows the mean excess plot and figure 4 shows the fitted GPD shape parameter vs the threshold. Based on these two figures, we can define our threshold to be 0.016. Using this threshold, the shape parameter is 0.30721 and the scale parameter is 0.00735. The PoT based VaR is 0.01553 and ES is 0.026362. Indeed, this is much more consistent with the historical data than the parametric or simulated method. The PoT analysis provided an additional layer of insight into extreme events. By isolating returns above user-defined thresholds, the analysis calculated VaR and ES with a focus on the most impactful losses. This method demonstrated its utility in capturing risks that traditional approaches might overlook, particularly for assets with significant deviations from normality. Figure 5 shows QQ plot for fitted GPD and figure 6 shows empirical and analytical CDF.

VaR Historical	-0.015341669	VaR Simulated	-0.01888
ES Historical	-0.026086761	ES Simulated	-0.02392
VaR Parametric	-0.017930162	Skewness	-0.16363
ES Parametric:	-0.022590441	Kurtosis	8.565153
VaR PoT	-0.015539455	Hill Estimator	2.309831
ES PoT	-0.026362432		

Table 5. Portfolio distribution analysis

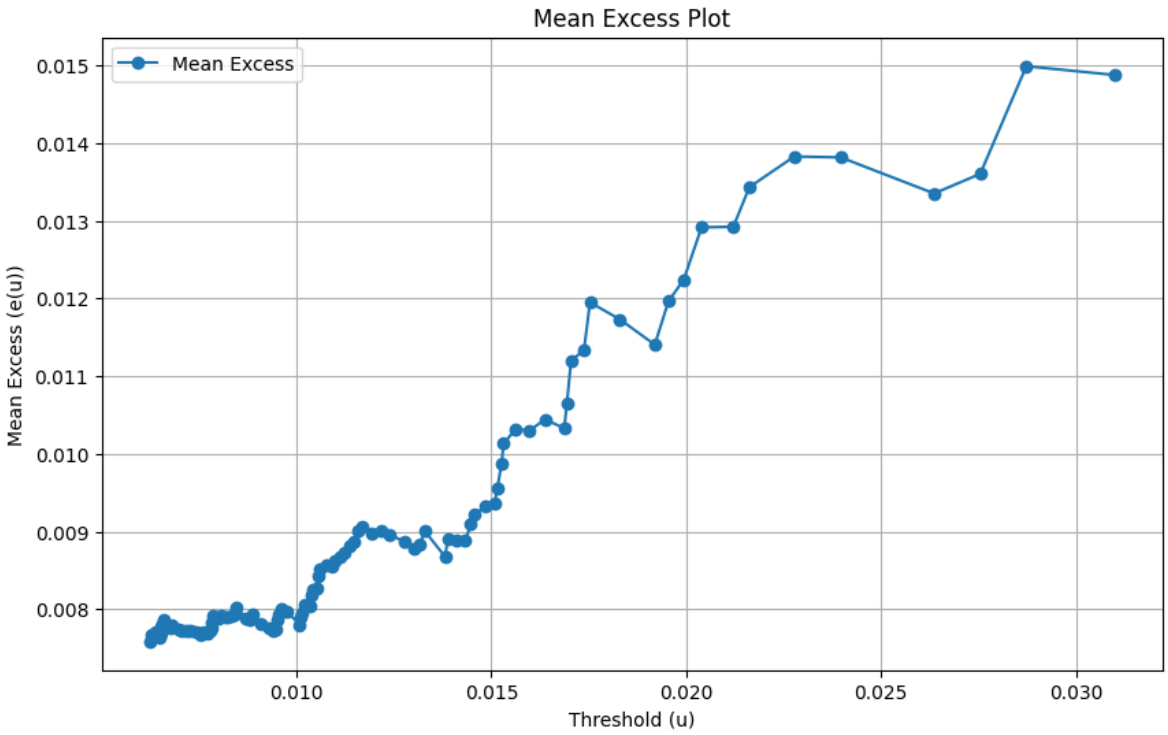


Figure 3. Mean excess plot VS Threshold

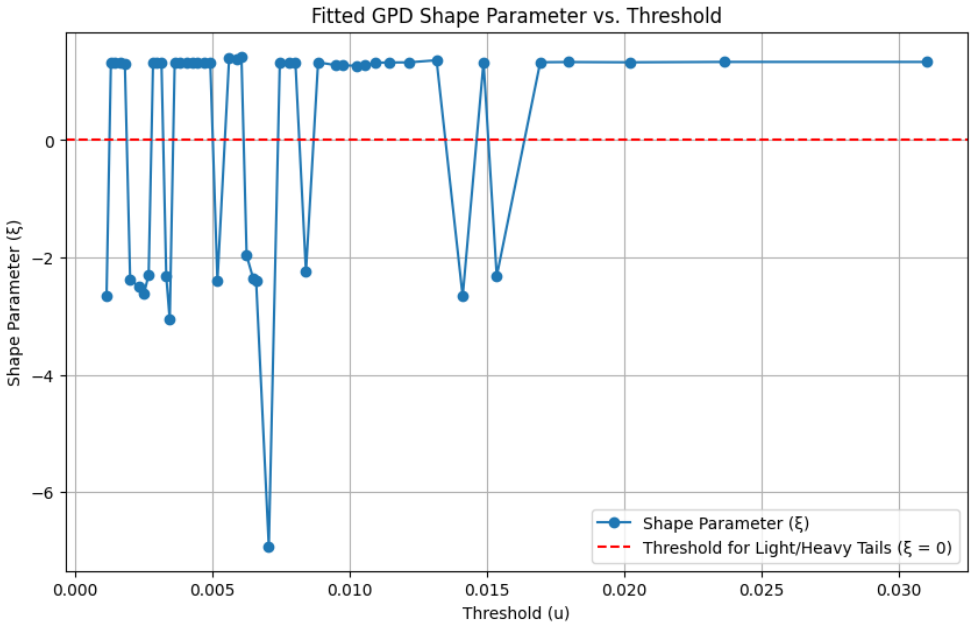


Figure 4. Fitted GPD shape parameter VS Threshold

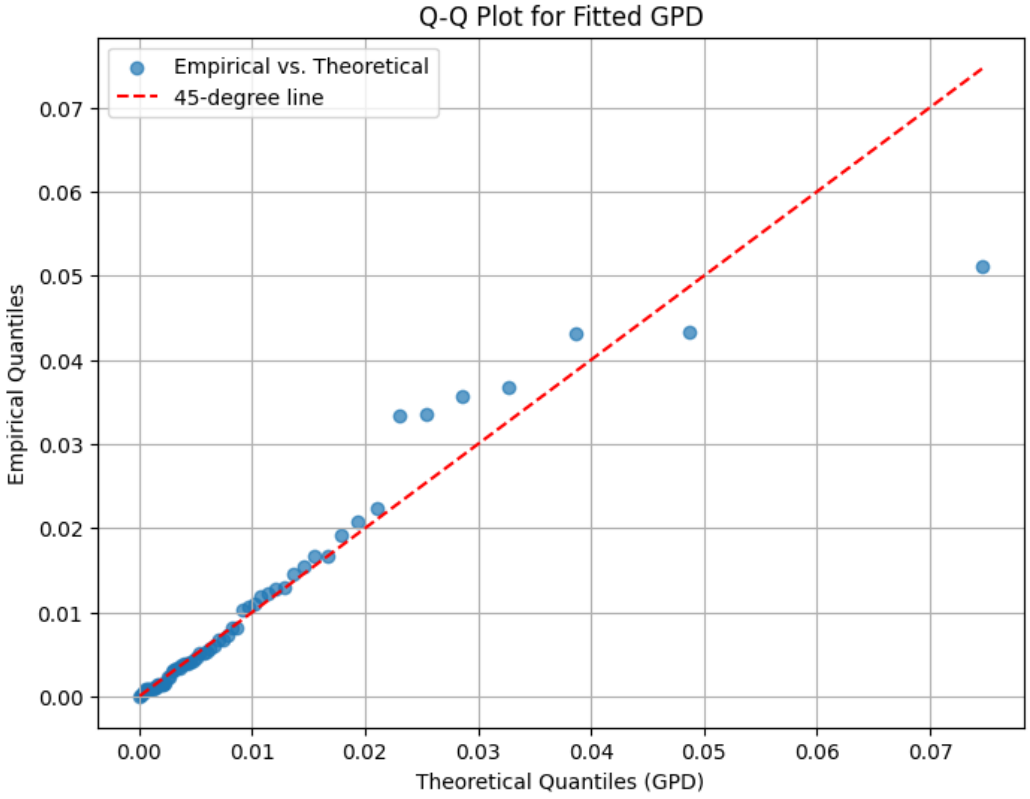


Figure 5. QQ plot for fitted GPD

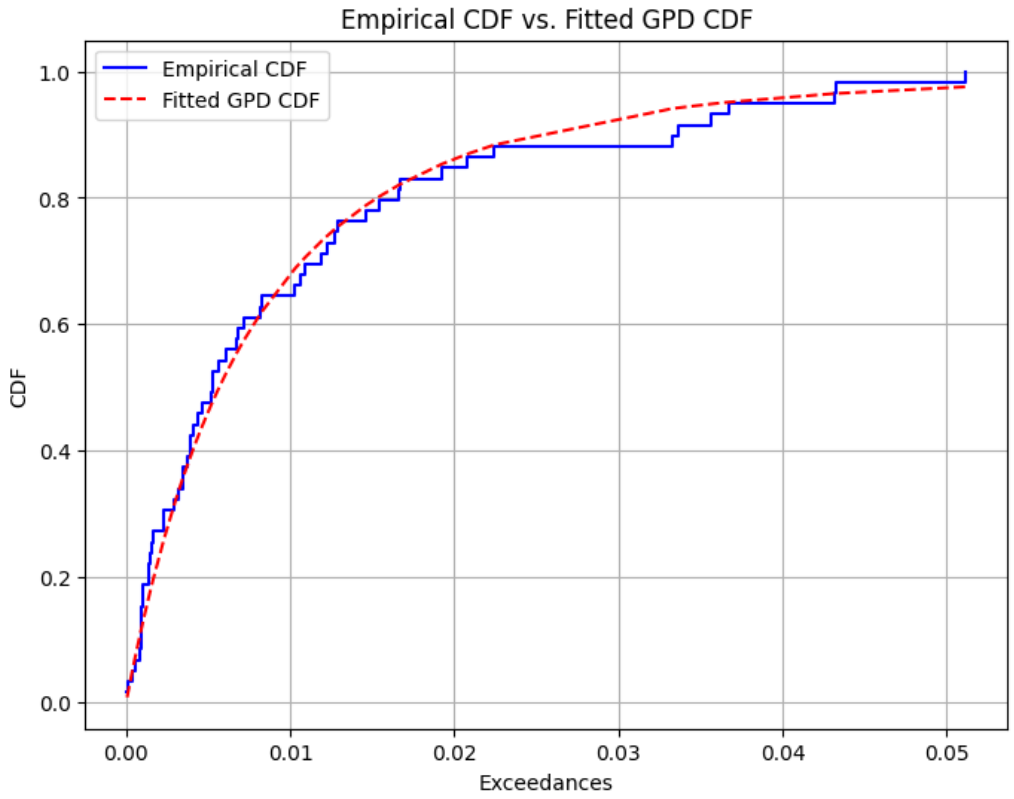


Figure 6. Empirical CDF vs Fitted GPD CDF

Conclusion

This analysis provides a robust framework for financial risk evaluation and portfolio optimization, combining traditional metrics like Value at Risk and Expected Shortfall with advanced techniques from Extreme Value Theory. Stress testing during periods such as the 2008 Financial Crisis highlighted the portfolio's vulnerabilities, while EVT methods, including Peaks Over Threshold and Generalized Pareto Distribution fitting, revealed significant tail risks. The PoT-based VaR and ES offered superior accuracy in capturing extreme losses compared to parametric and simulated methods.

Portfolio optimization minimized risk effectively, with the minimum variance portfolio achieving the highest Sharpe ratio, and the expected shortfall optimization prioritizing downside risk mitigation. Visualization tools such as mean excess plots and QQ plots validated the accuracy of the models. Overall, the framework underscores the importance of advanced risk modeling to better capture tail risks and improve portfolio resilience under both typical and extreme market conditions. Future research could explore alternative tail distributions such as Cauchy or exponential distributions to further enhance risk management strategies.

References

[1] McNeil, Alexander J., Rüdiger Frey, and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton University Press, 2015.

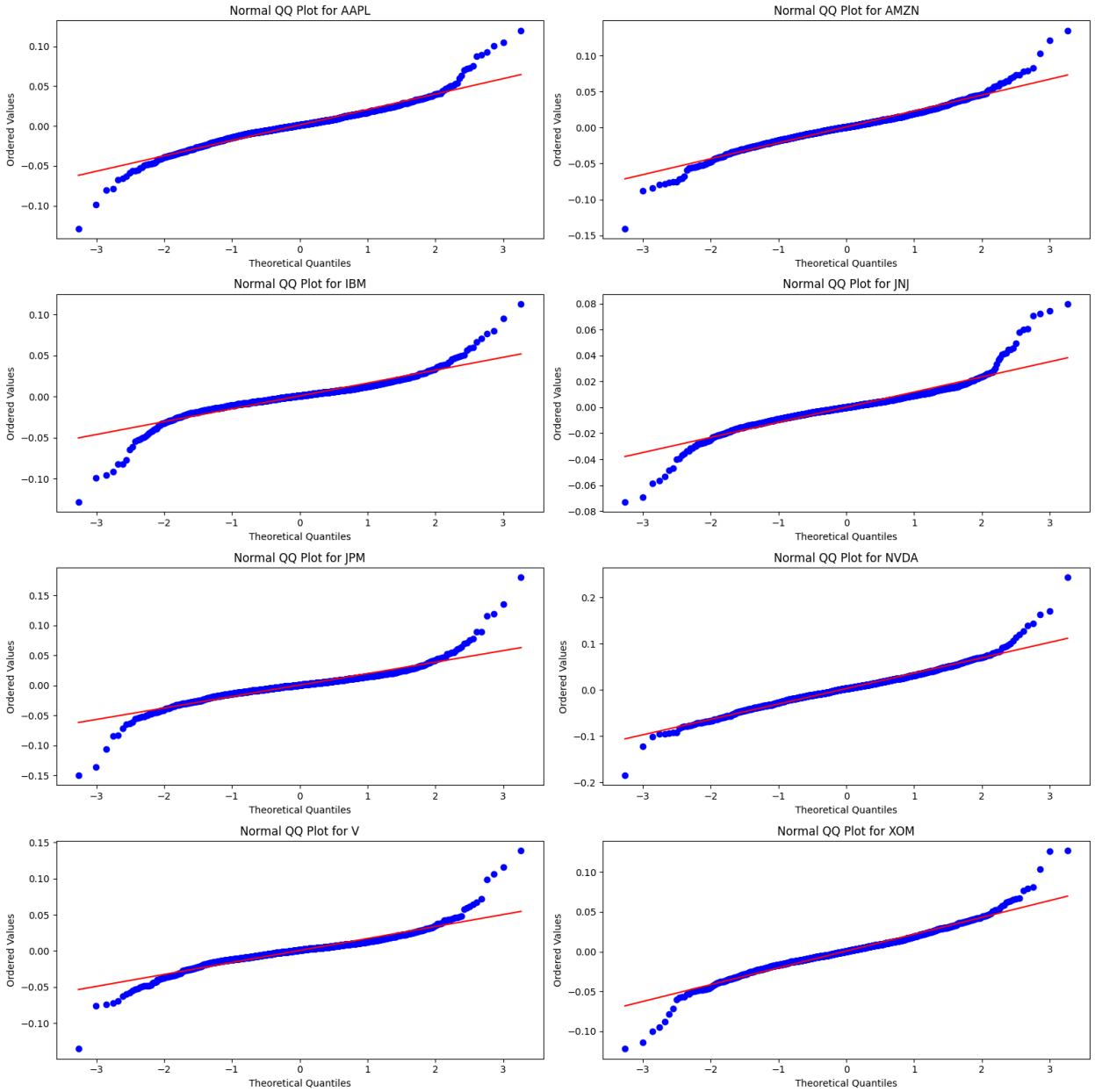
[2] Embrechts, Paul, Sidney I. Resnick, and Gennady Samorodnitsky. "Extreme value theory as a risk management tool." *North American Actuarial Journal* 3.2 (1999): 30-41.

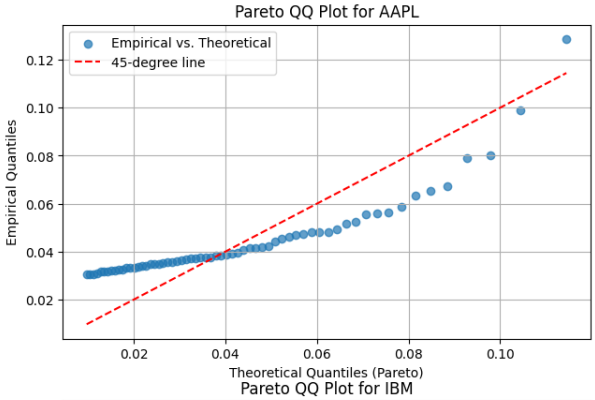
[3] Rockafellar, R. Tyrrell, and Stanislav Uryasev. "Optimization of conditional value-at-risk." *Journal of risk 2* (2000): 21-42.

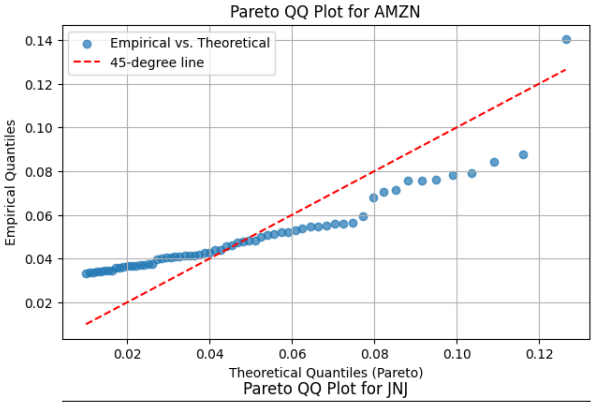
[4] Artzner, Philippe, et al. "Coherent measures of risk." *Mathematical finance* 9.3 (1999): 203-228.

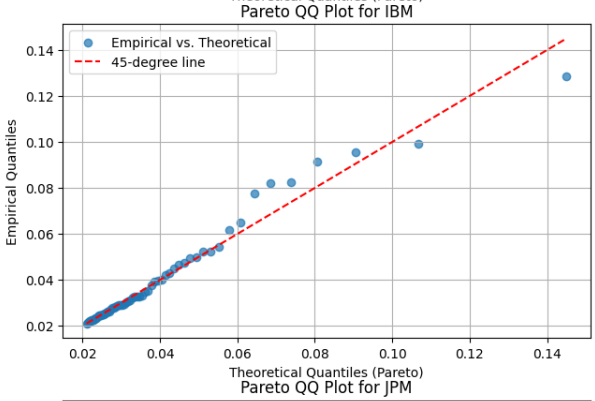
[5] Hull, John, and Alan White. "Incorporating volatility updating into the historical simulation method for value-at-risk." *Journal of risk* 1.1 (1998): 5-19.

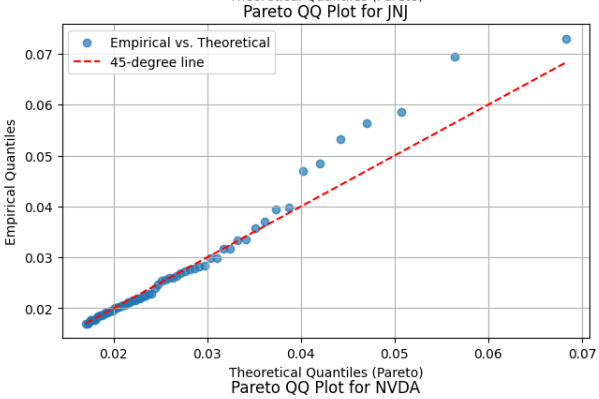
Appendix A: QQ plots

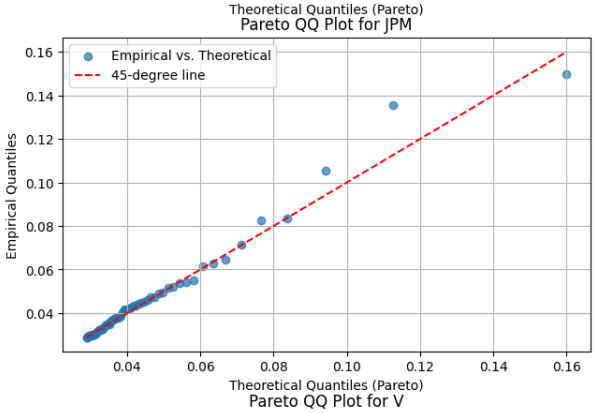


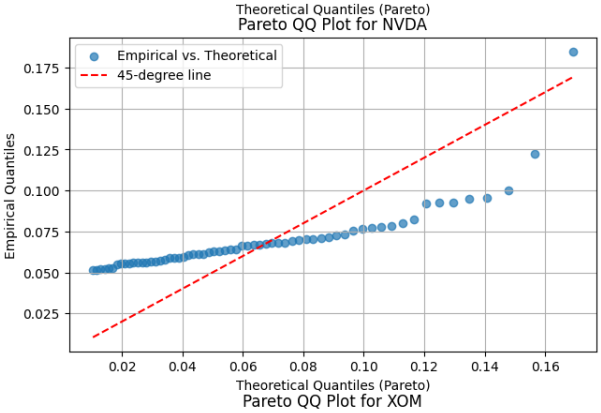

Pareto QQ Plot for AAPL
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)

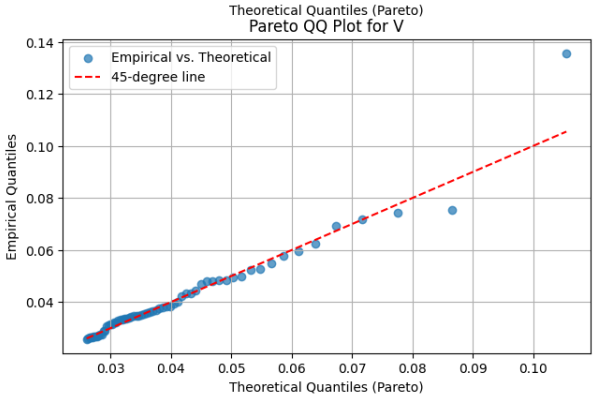

Pareto QQ Plot for AMZN
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)

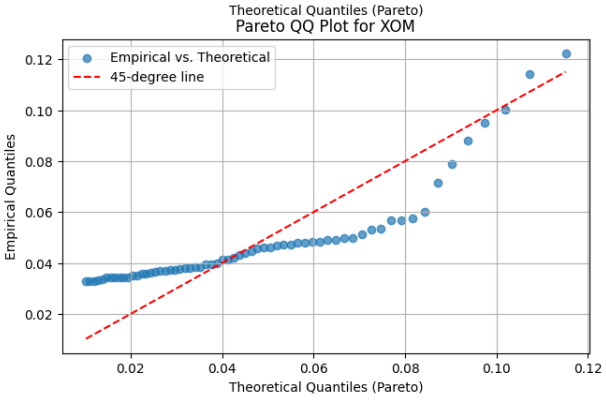

Pareto QQ Plot for IBM
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)


Pareto QQ Plot for JNJ
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)


Pareto QQ Plot for JPM
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)


Pareto QQ Plot for NVDA
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)


Pareto QQ Plot for V
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)


Pareto QQ Plot for XOM
Empirical vs. Theoretical
45-degree line
Empirical Quantiles
Theoretical Quantiles (Pareto)

Appendix B: Code

```
import pandas as pd
import numpy as np
import yfinance as yf
from scipy.stats import norm, genpareto, probplot, skew, kurtosis
import matplotlib.pyplot as plt
import scipy.stats as stats
import cvxopt as opt
from cvxopt import blas, solvers
import seaborn as sns

tickers = ["IBM", "NVDA", "AMZN", "V", "AAPL", "JPM", "JNJ", "XOM"]
end_date='2024-12-06'
start_date ='2019-12-06'

data = yf.download(
    tickers=tickers,
    start=start_date,
    end=end_date,
    #period='5y',
    interval='1d',
    prepost=False
)['Adj Close']

returns = data.pct_change().dropna()
confidence_level = 0.95
alpha = 1 - confidence_level
z_score = norm.ppf(confidence_level)

# Calculate VaR at the specified confidence level
VaR_hist = returns.quantile(alpha, axis=0)
# Calculate Expected Shortfall (ES)
ES_hist = returns[returns <= VaR_hist].mean()
print("Value at Risk (VaR) at 95% confidence level:")
print(VaR_hist)
print("\nExpected Shortfall (ES):")
print(ES_hist)

# Variance-Covariance VaR
mean_returns = returns.mean()
std_devs = returns.std()      #
# Calculate VaR
VaR_varcov = - (mean_returns + z_score * std_devs)
# Expected Shortfall (ES) using Variance-Covariance method
ES_varcov = - mean_returns + (std_devs * norm.pdf(z_score) / (1 - confidence_level))
print("\nVariance-Covariance VaR at 95% confidence level:")
print(VaR_varcov)
print("\nVariance-Covariance Expected Shortfall (ES):")
print(-ES_varcov)

num_simulations = 10000
monte_carlo_results = {}
for ticker in tickers:
    historical_returns = returns[ticker]
    mean = historical_returns.mean()
    std = historical_returns.std()

    simulated_returns = np.random.normal(mean, std, num_simulations)

    VaR_mc = np.percentile(simulated_returns, (1 - confidence_level) * 100)

    ES_mc = simulated_returns[simulated_returns <= VaR_mc].mean()
```

```
    monte_carlo_results[ticker] = {
        "VaR (Monte Carlo)": VaR_mc,
        "ES (Monte Carlo)": ES_mc
    }
mc_results_df = pd.DataFrame(monte_carlo_results).T
print("\nMonte Carlo Simulation Results:")
print(mc_results_df)

##=============================================================================================================================
==============================================##

# Stressed Period Data
stressed_periods = {
    "2008 Financial Crisis": ("2008-09-01", "2009-03-31"),
    "COVID-19 Pandemic": ("2020-02-01", "2020-04-30")
}
VaR_stressed_hist = {}
ES_stressed_hist = {}
VaR_stressed_varcov = {}
ES_stressed_varcov = {}

for period_name, (start_date, end_date) in stressed_periods.items():
    data_stressed = yf.download(
        tickers=tickers,
        start=start_date,
        end=end_date,
        interval='1d',
        prepost=False
    )['Adj Close']

    data_stressed = data_stressed.dropna()
    returns_stressed = data_stressed.pct_change().dropna()

    # Historical VaR and ES using stressed period data
    VaR_stressed_hist[period_name] = returns_stressed.quantile(alpha, axis=0)
    ES_stressed_hist[period_name] = returns_stressed[returns_stressed <= VaR_stressed_hist[period_name]].mean()

    # Variance-Covariance VaR using stressed parameters
    mean_returns_stressed = returns_stressed.mean()
    std_devs_stressed = returns_stressed.std()
    VaR_stressed_varcov[period_name] = - (mean_returns_stressed + z_score * std_devs_stressed)
    ES_stressed_varcov[period_name] = - mean_returns_stressed + (std_devs_stressed * norm.pdf(z_score) / (1 - confidence_level))

for period_name in stressed_periods.keys():
    print(f"\n--- {period_name} ---")
    print("Historical VaR at 95% confidence level:")
    print(VaR_stressed_hist[period_name])
    print("\nHistorical Expected Shortfall (ES):")
    print(ES_stressed_hist[period_name])
    print("\nVariance-Covariance VaR at 95% confidence level:")
    print(VaR_stressed_varcov[period_name])
    print("\nVariance-Covariance Expected Shortfall (ES):")
    print(ES_stressed_varcov[period_name])

##=============================================================================================================================
==============================================##

# Calculate mean returns and covariance matrix
mean_returns = returns.mean()
cov_matrix = returns.cov()
risk_free_rate = 0.04 / 365

# Portfolio optimization functions
def minimum_variance_portfolio(mean_returns, cov_matrix):
    n = len(mean_returns)
```

```
    S = opt.matrix(cov_matrix.values)
    G = None
    h = None
    A = opt.matrix(1.0, (1, n))
    b = opt.matrix(1.0)

    solvers.options['show_progress'] = False
    min_vol_solution = solvers.qp(S, opt.matrix(0.0, (n, 1)), G, h, A, b)
    min_vol_weights = np.array(min_vol_solution['x']).flatten()

    return min_vol_weights

min_vol_weights = minimum_variance_portfolio(mean_returns, cov_matrix)
print("\nMinimum Volatility Portfolio Weights:")
print(pd.Series(min_vol_weights, index=mean_returns.index))

# Calculate portfolio performance
def portfolio_performance(weights, mean_returns, cov_matrix, risk_free_rate):
    portfolio_return = np.dot(weights, mean_returns)
    portfolio_std_dev = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    sharpe_ratio = (portfolio_return - risk_free_rate) / portfolio_std_dev
    return portfolio_return, portfolio_std_dev, sharpe_ratio

min_vol_perf = portfolio_performance(min_vol_weights, mean_returns, cov_matrix, risk_free_rate)
print("\nMinimum Volatility Portfolio Performance (Return, Volatility, Sharpe):")
print(min_vol_perf)
print("\nOptimized portfolio weights:")
print(min_vol_weights)

def expected_shortfall_optimization(returns, confidence_level=0.95):
    n = returns.shape[1]
    T = returns.shape[0]
    alpha = 1 - confidence_level

    total_vars = n + 1 + T

    c = np.zeros(total_vars)
    c[n] = 1
    c[n+1:] = 1 / (alpha * T)
    c = opt.matrix(c)
    G = np.zeros((T, total_vars))
    G[:, :n] = -returns.values
    G[:, n] = -1
    G[:, n+1:] = -np.eye(T)
    G = opt.matrix(G)
    h = opt.matrix(np.zeros(T))
    G_s = np.zeros((T, total_vars))
    G_s[:, n+1:] = -np.eye(T)  # -s_i <= 0
    G_s = opt.matrix(G_s)
    h_s = opt.matrix(np.zeros(T))
    G = opt.matrix(np.vstack([G, G_s]))
    h = opt.matrix(np.vstack([h, h_s]))
    A = np.zeros((1, total_vars))
    A[0, :n] = 1
    A = opt.matrix(A)
    b = opt.matrix(1.0)

    # Solve optimization problem
    solvers.options['show_progress'] = False
    solution = solvers.lp(c, G, h, A, b)
    weights = np.array(solution['x'][:n]).flatten()

    return weights

# Optimize portfolio to minimize Expected Shortfall
es_weights = expected_shortfall_optimization(returns, confidence_level)
```

```
print("\nExpected Shortfall Minimization Portfolio Weights:")
print(pd.Series(es_weights, index=mean_returns.index))

# Calculate performance of ES optimized portfolio
es_portfolio_perf = portfolio_performance(es_weights, mean_returns, cov_matrix, risk_free_rate)
print("\nExpected Shortfall Minimization Portfolio Performance (Return, Volatility, Sharpe):")
print(es_portfolio_perf)

##============================================================================================================================================##

# Calculate skewness and kurtosis
skewness = returns.skew()
kurt = returns.kurtosis()

print("\nSkewness:\n", skewness)
print("\nKurtosis:\n", kurt)

# Calculate portfolio returns
portfolio_returns = returns.dot(min_vol_weights)

# Portfolio-level VaR
VaR_portfolio = portfolio_returns.quantile(alpha)

# Plot portfolio return distribution with VaR
sns.histplot(portfolio_returns, kde=True, bins=50, color='blue')
plt.axvline(VaR_portfolio, color='red', linestyle='dashed', linewidth=2, label='VaR')
plt.title("Minimum Variance Portfolio Return Distribution with VaR")
plt.legend()
plt.show()

# ES optimized portfolio returns
es_portfolio_returns = returns.dot(es_weights)

# Portfolio-level VaR for ES optimized portfolio
VaR_es_portfolio = es_portfolio_returns.quantile(alpha)

# Plot ES optimized portfolio return distribution with VaR
sns.histplot(es_portfolio_returns, kde=True, bins=50, color='green')
plt.axvline(VaR_es_portfolio, color='red', linestyle='dashed', linewidth=2, label='VaR')
plt.title("ES Optimized Portfolio Return Distribution with VaR")
plt.legend()
plt.show()

# Tail ratio calculation per ticker
top_5 = returns.apply(lambda x: x[x >= x.quantile(0.95)].mean())
bottom_5 = returns.apply(lambda x: x[x <= x.quantile(0.05)].mean())
tail_ratio = top_5 / abs(bottom_5)
print("\nTail Ratio (Top 5% / Bottom 5%):")
print(tail_ratio)

returns_clean = returns.replace([np.inf, -np.inf], np.nan).dropna()

hill_estimators = {}
for ticker in returns_clean.columns:
    returns_ticker = returns_clean[ticker]
    threshold = returns_ticker.quantile(0.95)
    extreme_returns = returns_ticker[returns_ticker > threshold]

    if len(extreme_returns) > 1:
        sorted_extremes = np.sort(extreme_returns)[::-1]

        k = len(sorted_extremes) - 1
        dynamic_threshold = sorted_extremes[-1]
        hill_estimator = (1 / k) * sum(np.log(sorted_extremes[:-1] / dynamic_threshold))
        hill_estimators[ticker] = 1/hill_estimator
```

```
    else:
        hill_estimators[ticker] = np.nan

hill_estimators_series = pd.Series(hill_estimators)
print("\nHill Estimator for Tail Risk:\n", hill_estimators_series)

n_rows, n_cols = 4, 2
fig_normal, axes_normal = plt.subplots(n_rows, n_cols, figsize=(16, 16))
axes_normal = axes_normal.flatten()

fig_pareto, axes_pareto = plt.subplots(n_rows, n_cols, figsize=(16, 20))
axes_pareto = axes_pareto.flatten()
for i, ticker in enumerate(returns_clean.columns):
    ticker_returns = returns_clean[ticker]

    # Normal QQ Plot
    if len(ticker_returns) > 0:
        probplot(ticker_returns, dist="norm", plot=axes_normal[i])
        axes_normal[i].set_title(f"Normal QQ Plot for {ticker}")
        axes_normal[i].set_xlabel("Theoretical Quantiles")
        axes_normal[i].set_ylabel("Ordered Values")
    else:
        print(f"No data available to plot Normal QQ plot for {ticker}.")

    # Pareto QQ Plot
    threshold = ticker_returns.quantile(0.05)
    left_tail = ticker_returns[ticker_returns <= threshold]

    if not np.all(np.isfinite(left_tail)):
        print(f"Non-finite values detected in left-tail data for {ticker}.")
        left_tail = left_tail[np.isfinite(left_tail)]

    neg_left_tail = -left_tail.values.flatten()
    if len(neg_left_tail) > 0:
        params = genpareto.fit(neg_left_tail)
        shape, loc, scale = params
        theoretical_quantiles = genpareto.ppf(
            np.linspace(0.01, 0.99, len(neg_left_tail)), shape, loc=loc, scale=scale
        )
        sorted_empirical = np.sort(neg_left_tail)

        # Pareto QQ Plot
        axes_pareto[i].scatter(theoretical_quantiles, sorted_empirical, alpha=0.7, label="Empirical vs. Theoretical")
        axes_pareto[i].plot(theoretical_quantiles, theoretical_quantiles, color='red', linestyle='--', label="45-degree line")
        axes_pareto[i].set_title(f"Pareto QQ Plot for {ticker}")
        axes_pareto[i].set_xlabel("Theoretical Quantiles (Pareto)")
        axes_pareto[i].set_ylabel("Empirical Quantiles")
        axes_pareto[i].legend()
        axes_pareto[i].grid()
    else:
        print(f"Not enough data in the left tail for Pareto QQ plot for {ticker}.")

fig_normal.tight_layout()
plt.show()
fig_pareto.tight_layout()
plt.show()

##=========================================================================================================================
==========================================##

# Extreme Risk Analysis on an Optimized Portfolio
portfolio_returns = -returns.dot(min_vol_weights)

# Portfolio VaR and ES
VaR_portfolio_hist = portfolio_returns.quantile(alpha)
ES_portfolio_hist = portfolio_returns[portfolio_returns <= VaR_portfolio_hist].mean()
```

```
# Variance-Covariance VaR and ES for the Portfolio
portfolio_mean_return = portfolio_returns.mean()
portfolio_std_dev = portfolio_returns.std()
VaR_portfolio_varcov = -(portfolio_mean_return + z_score * portfolio_std_dev)
ES_portfolio_varcov = -(portfolio_mean_return + portfolio_std_dev * norm.pdf(z_score) / (1 - confidence_level))

# Monte Carlo VaR and ES for the Portfolio
simulated_portfolio_returns = np.random.normal(portfolio_mean_return, portfolio_std_dev, num_simulations)
VaR_portfolio_mc = np.percentile(simulated_portfolio_returns, (1 - confidence_level) * 100)
ES_portfolio_mc = simulated_portfolio_returns[simulated_portfolio_returns <= VaR_portfolio_mc].mean()

# Tail Statistics
portfolio_skewness = skew(portfolio_returns)
portfolio_kurtosis = kurtosis(portfolio_returns)

# Hill Estimator
portfolio_extreme_returns = portfolio_returns[portfolio_returns > portfolio_returns.quantile(0.95)]

if len(portfolio_extreme_returns) > 1:
    sorted_extremes = np.sort(portfolio_extreme_returns)[::-1]

    dynamic_threshold = sorted_extremes[-1]
    k = len(sorted_extremes) - 1
    hill_estimator_portfolio = 1/((1 / k) * sum(np.log(sorted_extremes[:-1] / dynamic_threshold)))
else:
    hill_estimator_portfolio = np.nan

portfolio_results = {
    "VaR_Historical": VaR_portfolio_hist,
    "ES_Historical": ES_portfolio_hist,
    "VaR_VarianceCovariance": VaR_portfolio_varcov,
    "ES_VarianceCovariance": ES_portfolio_varcov,
    "VaR_MonteCarlo": VaR_portfolio_mc,
    "ES_MonteCarlo": ES_portfolio_mc,
    "Skewness": portfolio_skewness,
    "Kurtosis": portfolio_kurtosis,
    "Hill_Estimator": hill_estimator_portfolio,
}

print("\nPortfolio-Level Extreme Risk Analysis Results:")
for key, value in portfolio_results.items():
    print(f"{key}: {value}")

##=============================================================================================================================
=============================================##

# Mean Excess Plot Function
def mean_excess_plot(data, quantiles=None):
    data = np.sort(data)

    if quantiles is None:
        quantiles = np.linspace(0.8, 0.99, 100)
    thresholds = np.quantile(data, quantiles)

    mean_excess_values = []
    for u in thresholds:
        exceedances = data[data > u] - u
        if len(exceedances) > 0:
            mean_excess_values.append(exceedances.mean())
        else:
            mean_excess_values.append(np.nan)

    plt.figure(figsize=(10, 6))
    plt.plot(thresholds, mean_excess_values, marker='o', linestyle='-', label="Mean Excess")
    plt.xlabel("Threshold (u)")
```

```
    plt.ylabel("Mean Excess (e(u))")
    plt.title("Mean Excess Plot")
    plt.grid(True)
    plt.legend()
    plt.show()

# GPD Shape Parameter vs. Threshold Plot
def gpd_shape_vs_threshold(data, quantiles=None):
    data = np.sort(data)

    if quantiles is None:
        quantiles = np.linspace(0.6, 0.99, 50)
    thresholds = np.quantile(data, quantiles)

    shape_parameters = []

    for u in thresholds:
        exceedances = data[data > u] - u
        if len(exceedances) > 0:
            params = genpareto.fit(-exceedances)
            shape, _, _ = params
            shape_parameters.append(shape)
        else:
            shape_parameters.append(np.nan)

    plt.figure(figsize=(10, 6))
    plt.plot(thresholds, shape_parameters, marker='o', linestyle='-', label="Shape Parameter (\u03BE)")
    plt.axhline(0, color='red', linestyle='--', label="Threshold for Light/Heavy Tails (\u03BE = 0)")
    plt.xlabel("Threshold (u)")
    plt.ylabel("Shape Parameter (\u03BE)")
    plt.title("Fitted GPD Shape Parameter vs. Threshold")
    plt.grid(True)
    plt.legend()
    plt.show()

mean_excess_plot(portfolio_returns)
gpd_shape_vs_threshold(portfolio_returns)


def pot_var_es(data, threshold, confidence_level):

    exceedances = data[data > threshold] - threshold

    if len(exceedances) == 0:
        raise ValueError("No exceedances above the threshold. Choose a lower threshold.")

    shape, loc, scale = genpareto.fit(exceedances, floc=0)

    num_exceedances = len(exceedances)
    total_data_points = len(data)
    probability_exceedance = num_exceedances / total_data_points

    adjusted_probability = (1 - confidence_level) / probability_exceedance
    VaR = threshold + (scale / shape) * ((adjusted_probability) ** (-shape) - 1)

    if shape < 1:
        ES = VaR + (scale - shape * (VaR - threshold)) / (1 - shape)
    else:
        ES = np.inf

    return VaR, ES

threshold = 0.016
confidence_level = 0.95

# Calculate VaR and ES using PoT
```

```
try:
    VaR_pot, ES_pot = pot_var_es(portfolio_returns, threshold, confidence_level)
    print(f"PoT VaR at {confidence_level * 100}% confidence: {VaR_pot}")
    print(f"PoT ES at {confidence_level * 100}% confidence: {ES_pot}")
except ValueError as e:
    print(e)


def fit_gpd_and_visualize(data, threshold):
    exceedances = data[data > threshold] - threshold

    if len(exceedances) == 0:
        raise ValueError("No exceedances above the threshold. Choose a lower threshold.")

    shape, loc, scale = genpareto.fit(exceedances, floc=0)
    params = (shape, loc, scale)
    print(f"Fitted GPD Parameters: Shape={shape}, Location={loc}, Scale={scale}")

    theoretical_quantiles = genpareto.ppf(
        np.linspace(0.01, 0.99, len(exceedances)), shape, loc=loc, scale=scale
    )
    empirical_quantiles = np.sort(exceedances)

    plt.figure(figsize=(8, 6))
    plt.scatter(theoretical_quantiles, empirical_quantiles, alpha=0.7, label="Empirical vs. Theoretical")
    plt.plot(theoretical_quantiles, theoretical_quantiles, color='red', linestyle='--', label="45-degree line")
    plt.title("Q-Q Plot for Fitted GPD")
    plt.xlabel("Theoretical Quantiles (GPD)")
    plt.ylabel("Empirical Quantiles")
    plt.legend()
    plt.grid()
    plt.show()

    sorted_exceedances = np.sort(exceedances)
    ecdf = np.arange(1, len(sorted_exceedances) + 1) / len(sorted_exceedances)  # Empirical CDF
    fitted_cdf = genpareto.cdf(sorted_exceedances, shape, loc=loc, scale=scale)  # Fitted GPD CDF

    plt.figure(figsize=(8, 6))
    plt.step(sorted_exceedances, ecdf, where='post', label="Empirical CDF", color="blue")
    plt.plot(sorted_exceedances, fitted_cdf, label="Fitted GPD CDF", color="red", linestyle="--")
    plt.title("Empirical CDF vs. Fitted GPD CDF")
    plt.xlabel("Exceedances")
    plt.ylabel("CDF")
    plt.legend()
    plt.grid()
    plt.show()

    return params

threshold = 0.016
params = fit_gpd_and_visualize(portfolio_returns, threshold)
```