Gabe
GabeL@virginia.edu
https://gabriel-vzh2vs.github.io/SYS3062-Website/

UVA's Systems and Information Engineering

Feburary 3rd, 2026

SYS 3062: Lab 3

Portfolio With Confidence: Value-At-Risk

Monte Carlo for Portfolios

Today's Objective

We will apply Monte Carlo simulation to a dataset of stock prices.

- Builds directly on last week's skills.
- Moves from theoretical models to real-world financial data.

Expectations & Guidance

Since this is our second week with Monte Carlo:

- **Focus on Application:** Only One Mandatory Mathematical Concept will be introduced today.

Legal Disclaimer

Educational Purposes Only. Nothing in this presentation should be interpreted or acted upon as financial advice.

GabeL@virginia.edu

To simulate a portfolio correctly, we cannot simply run separate simulations for each stock and add them up. We need the stocks to move in sync with their historical relationships.

The Cholesky Decomposition

We use the **Covariance Matrix** of returns (Σ) to generate correlated random variables.

$$R_{correlated} = \mu + L \cdot Z$$

- Z: A vector of uncorrelated random numbers $\sim N(0, 1)$.
- L: The lower triangular matrix from Cholesky decomposition ($\Sigma = LL^T$).
- Result: Returns that "move together" like the real market.

‣ Want More Details?

GabeL@virginia.edu

Part 1: Scenario & Execution

The Scenario

You are an investment analyst managing a diverse portfolio of eight stocks/funds:

- **Tickers:** MSFT, TMUS, V, A, GOOGL, SPOT, VXX, C.
- **Objective:** Identify and quantify the specific risk factors driving this portfolio.

Execution Steps (Intermediate Tasks)

1. **Data Acquisition:** Obtain market data for the listed stocks.
2. **Statistical Baseline:** Calculate the Standard Deviation and Variance for both *Prices* and *Returns*.
3. **Weighting:** Allocation the initial portfolio value according to specific weights.
4. **Simulation:** Simulate **one month** of returns based on the historical data parameters.

GabeL@virginia.edu

Part 2: Required Analysis

Based on your simulation, you must formulate conclusions on the following key risk areas:

Risk Metrics

- **Value-at-Risk (VaR):** Determine potential losses using Confidence Intervals, $\alpha = 0.95$.
- **Variance Contribution:** Which specific stocks are contributing the most to the total portfolio volatility?
- **Normality:** How does Sample Size affect the assumption of Normality?

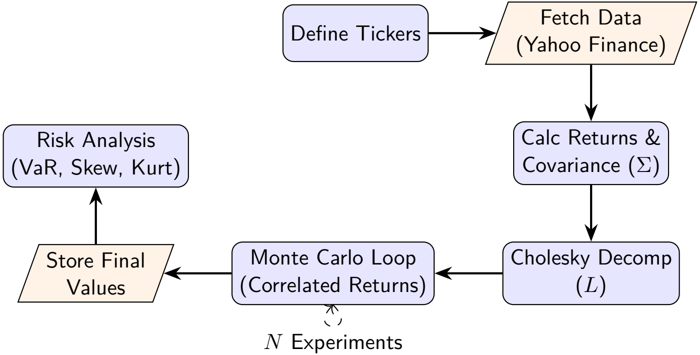
Scenario Insights

- **Extreme Events:** What are the risks of "tail events" (Skew/Kurtosis)?
- **Sensitivity:** What are the effects of changing the portfolio composition (weights) on the overall variance?

GabeL@virginia.edu

Lab 3 Process Flow

The simulation pipeline moves from data acquisition to statistical baseline, then simulation, and finally risk analysis.



Algorithm Design: Correlated Simulation

Algorithm 1 Portfolio Monte Carlo

$Prices \leftarrow$ FetchData$(Tickers)$
$Returns \leftarrow \ln(Prices_t / Prices_{t-1})$
$\Sigma \leftarrow$ Covariance$(Returns)$
$L \leftarrow$ Cholesky(Σ)
for $i = 1$ **to** $N_{experiments}$ **do**
    $Value \leftarrow P_0$
    for $d = 1$ **to** $Days$ **do**
        $Z \leftarrow$ RandomNormal$(0, 1, size = len(Tickers))$
        $R_{daily} \leftarrow \mu + L \times Z$ ▷ Correlate returns
        $R_{port} \leftarrow$ Dot$(Weights, R_{daily})$
        $Value \leftarrow Value \times e^{R_{port}}$
    end for
    Store $Value$
end for

GabeL@virginia.edu

Python Simulation Implementation

```python
def run_simulation(weights, mean_returns, cov_matrix,
p0, days, experiments):

    L = np.linalg.cholesky(cov_matrix)
    final_values = []
    for i in range(experiments):
        portfolio_value = p0
        for i in range(days):
            Z = np.random.normal(0, 1, len(weights))
            daily_returns = mean_returns.values + np.dot(L, Z)
            portfolio_return = np.dot(weights, daily_returns)
        portfolio_value = portfolio_value * np.exp(portfolio_return)
        final_values.append(portfolio_value)
    return np.array(final_values)
```

Risk Analysis: Value-at-Risk & Intervals

Once we have the simulated results, we analyze the distribution to understand the risk profile.

Value-at-Risk (VaR):

- The 5th percentile of outcomes.
- "We are 95% confident losses will not exceed this amount."

Confidence Intervals:

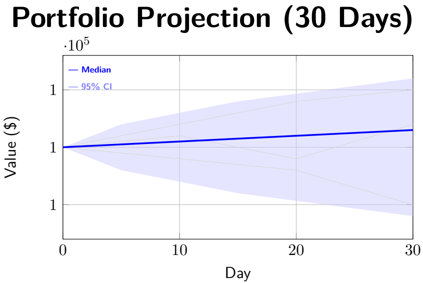
- We visualize the 95% envelope (Fan Chart).
- Shows the spread of possible futures.

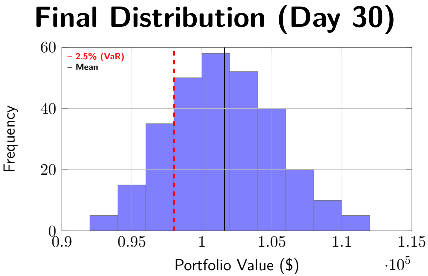
GabeL@virginia.edu

Python Code for Analysis

```
var_95 = np.percentile(returns, 5)
skewness = scipy.stats.skew(final_values)
kurtosis = scipy.stats.kurtosis(final_values)
```

These diagrams replicate the standard outputs from the Python simulation.





GabeL@virginia.edu

Your Turn

Now, it's your turn to implement Value at Risk in Python or using XLRisk! Good Luck!

GabeL@virginia.edu

A More Realistic Extension

The Problem with "1/N" Allocation

Previously in Lab 3, we used Equal Weighting (1/8th per stock).

- **Issue:** Risky assets (like VXX) contribute disproportionately to the portfolio's total volatility.
- **Result:** You aren't truly diversified; you are holding a "basket of volatility".

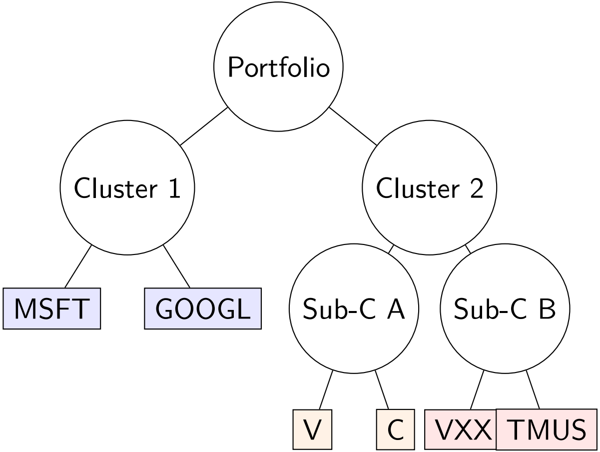
The Solution: Optimization Instead of guessing weights, we use algorithms to find the optimal balance.

- **Modern Portfolio Theory (MPT):** Standard Mean-Variance (often unstable).
- **Machine Learning Approach: HERC** (Hierarchical Equal Risk Contribution).

GabeL@virginia.edu

Concept: Hierarchical Clustering

HERC does not treat all assets as a flat list. It uses Clustering to group similar assets (e.g., Tech Stocks vs. Volatility Hedges) and allocates risk budgets recursively.



Assets are grouped by correlation (Linkage), then weights are assigned top-down.

GabeL@virginia.edu

Algorithm 2 HERC Optimization Pipeline

$Returns \leftarrow \text{PctChange}(Prices)$
$DistanceMatrix \leftarrow \sqrt{2(1 - \text{Correlation}(Returns))}$
$Clusters \leftarrow \text{HierarchicalClustering}(DistanceMatrix, \text{'ward'})$
$Weights \leftarrow []$
function $\textsc{RecursiveBisection}(\text{Node})$
    if Node is Leaf **then return**
    end if
    $LeftRisk \leftarrow \text{CalculateRisk}(Node.Left)$
    $RightRisk \leftarrow \text{CalculateRisk}(Node.Right)$
    $Allocratio \leftarrow 1 - \frac{LeftRisk}{LeftRisk+RightRisk}$
    AssignWeights based on Ratio
end function
$OptimalWeights \leftarrow \text{Run}(Clusters)$

GabeL@virginia.edu

Python Implementation (Riskfolio)

We define the portfolio model and optimize using the `HERC` method with Pearson correlation.

```python
!pip install riskfolio
import riskfolio as rp

def get_herc_weights(data):
    Y = data.pct_change().dropna()
    port = rp.HCPortfolio(returns=Y)

    # model='HERC': Hierarchical Equal Risk Contribution
    # linkage='ward': Minimizes variance within clusters
    weights_df = port.optimization(
        model='HERC',
        codependence='pearson',
        rm='MV',  # Risk Measure = Variance
        linkage='ward')
    return weights_df['weights'].values
```

Result: Optimized Risk Contribution

Unlike Equal Weighting, where risk is uneven, HERC strives to equalize the risk contribution of each asset cluster.

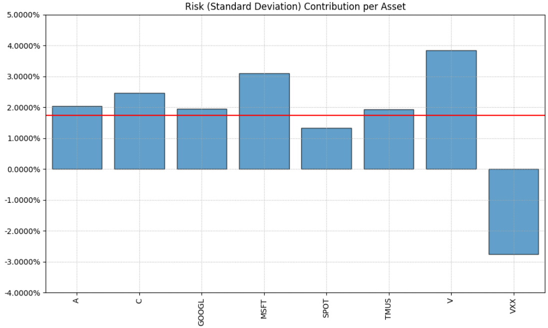


Figure: Risk Contribution

GabeL@virginia.edu

Optimization vs. Naive Allocation

Naive (Equal Weights)

- Easy to implement.
- Uses Cholesky Decomposition correlations.
- **Outcome:** Portfolio dominated by the most volatile stock (e.g., Tesla or VXX).

Optimized (HERC)

- Mathematically robust.
- Groups correlated assets.
- **Outcome:** Lower drawdown and smoother equity curve in Monte Carlo simulations.

GabeL@virginia.edu

Feedback Form

Throughout the course, I may ask for your feedback on:

- **Course Materials**
- **Lecture & Lab**
- **Assignments**

GabeL@virginia.edu

Action Items

And now, you need to do the following tasks:

- **Submit** your Python file to Gradescope by next week's lab;
- **Read** Project 1 in the textbook, and we will have a lab and then a workday next week;
- **Fill** Out the Group form below for Project 1.

GabeL@virginia.edu

Backup Slide: Python Image

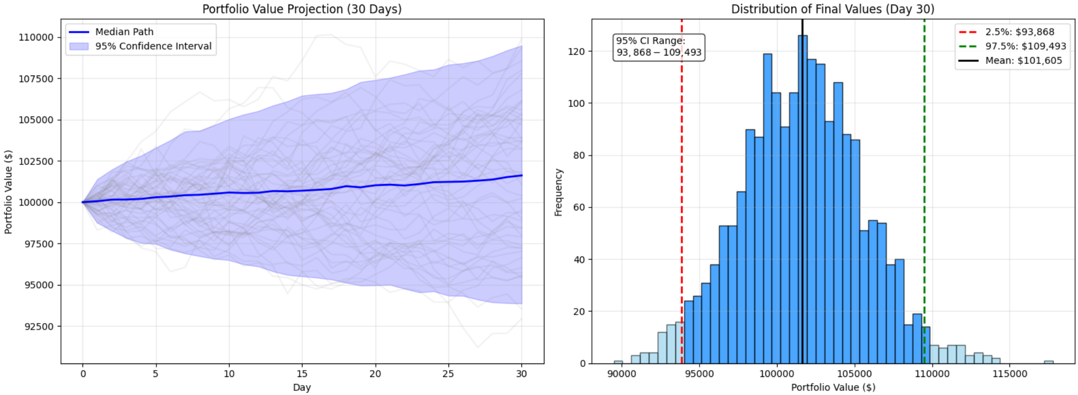


Figure: The Real Image from the Output

GabeL@virginia.edu

The Challenge: We want to simulate two assets (X_1, X_2) that are not independent. They must have a specific correlation of $\rho = \mathbf{0.7}$.

Step 1: Define the Target Matrix ($\mathbf{C}$)

We start by writing down the Correlation Matrix we want to achieve:

$$\mathbf{C} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0.7 \\ 0.7 & 1 \end{pmatrix}$$

Our goal is to find a "square root" matrix $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^T = \mathbf{C}$.

Step 2: Solve for $\mathbf{L}$

For a 2×2 matrix, the Cholesky matrix $\mathbf{L}$ always follows this specific form:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}$$

The Calculation: We only need to solve for the bottom-right term:

$$\sqrt{1-\rho^2} = \sqrt{1-(0.7)^2} = \sqrt{1-0.49} = \sqrt{0.51} \approx \mathbf{0.714}$$

The Resulting Matrix:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ 0.7 & 0.714 \end{pmatrix}$$

GabeL@virginia.edu

Now we use $\mathbf{L}$ to transform random noise into correlated data.

Step 3: Generate Independent Noise ($\mathbf{Z}$)

$$\text{Draw } Z_1, Z_2 \sim N(0,1) \quad \text{(e.g., } Z_1 = 0.5, Z_2 = -1.2\text{)}$$

Step 4: Apply the Transformation ($\mathbf{X} = \mathbf{LZ}$)

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.7 & 0.714 \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix}$$

$$\begin{aligned} X_1 &= \mathbf{Z_1} \\ X_2 &= 0.7(\mathbf{Z_1}) + 0.714(\mathbf{Z_2}) \end{aligned}$$

GabeL@virginia.edu