Data Analytics in Stock Market

Group 5: Kaichen Bian, Longtao Chen, Zeyu He, Siying Chen, Shuhong Cai

Stock Picking

Johnson & Johnson (JNJ): **Health Care** - Pharmaceuticals, Medical Devices, and Consumer Health Products.

Walmart (WMT): **Consumer Staples** - Retailing (specifically Hypermarkets & Super Centers).

Bank of America (BAC): **Financials** - Banks.

Microsoft (MSFT): **Information Technology** - Software & Services.

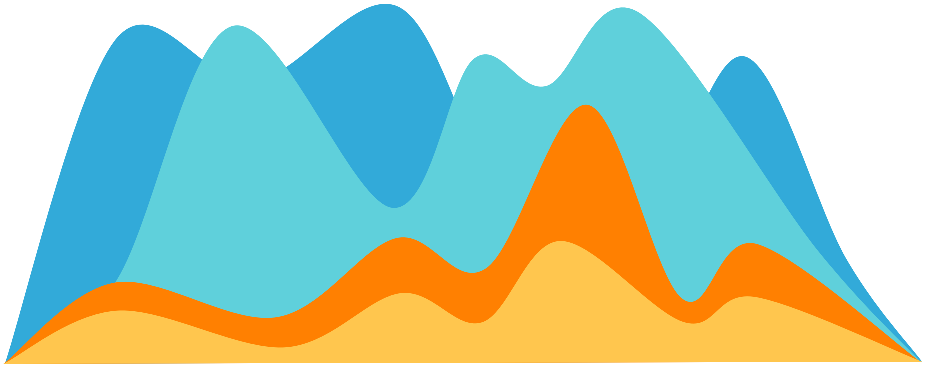
Contents

Target Customers

Stock Market

- 01 Investment Firms/Business Managers — More Informative
- 02 Portfolio Managers/Financial Analysts — Smarter
- 03 Individual Investors — More instant

Value Proposition



Human Intuition

Algorithmic Trading

pre-programmed rules to execute trades automatically

real-time data, historical trends, and complex models

faster, more precise, and removes emotion from the trading process.

Dataset Overview

- **Daily Return** dataset used in this project contains **5,000** records spanning from **2019-01-01** to **2023-12-29**.
- **FF5** from **1963-01** to **2024-07**
- **Financial Statement** contains **18** variables from **2019** to **2023**

This dataset was sourced from **Bloomberg, Yahoo Finance** and **Prof. French's site**.

Data Cleaning Process

Removing Missing Values

Missing values in key columns like Return, Volatility, Revenue, and Net Income were addressed by removing incomplete rows, ensuring a reliable and complete dataset for analysis.

Handling Outliers

Outliers in metrics such as Revenue, Net Income, and stock returns were removed using the IQR method, maintaining a dataset representative of typical performance.

Normalizing Data Types

Numeric fields treated as text were converted to proper numeric formats, standardizing data types for consistent and accurate analysis.

```
# Fama
# Step 1: Removing missing values
def remove_missing_values(df):
    return df.dropna()

# Step 2: Handling outliers (using IQR method)
def remove_outliers(df, columns):
    for column in columns:
        df.loc[:, column] = pd.to_numeric(df[column], errors='coerce')
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

# Step 3: Normalizing data types
def normalize_data_types(df, columns):
    for column in columns:
        df.loc[:, column] = pd.to_numeric(df[column], errors='coerce')
    return df

# Apply all steps in one process
columns_to_process = ['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']

# Clean Fama Factors data
cleaned_fama_factors = remove_missing_values(fama_factors_df)
cleaned_fama_factors_no_outliers = remove_outliers(cleaned_fama_factors, columns_to_process)
cleaned_fama_factors_normalized = normalize_data_types(cleaned_fama_factors_no_outliers, columns_to_process)

# Export the cleaned data to Desktop as Excel
output_path = '/Users/longtaochen/Desktop/Cleaned_Fama_Factors.xlsx'  # Replace with your correct path

# Write the cleaned dataset to Excel
cleaned_fama_factors_normalized.to_excel(output_path, sheet_name='Cleaned_Fama_Factors', index=False)

print(f"Fama Factors data has been exported to {output_path}")
```

Normalization

Sector ID	Sector Name
HIN	Heathcare
XLF	Financial
XLK	Techonology
XLP	Consumer Staples

1st Normalization:

- All tables are in a tabular format with atomic (indivisible) values
- E.g: *Sector_id, Sector_Name* in *sector*

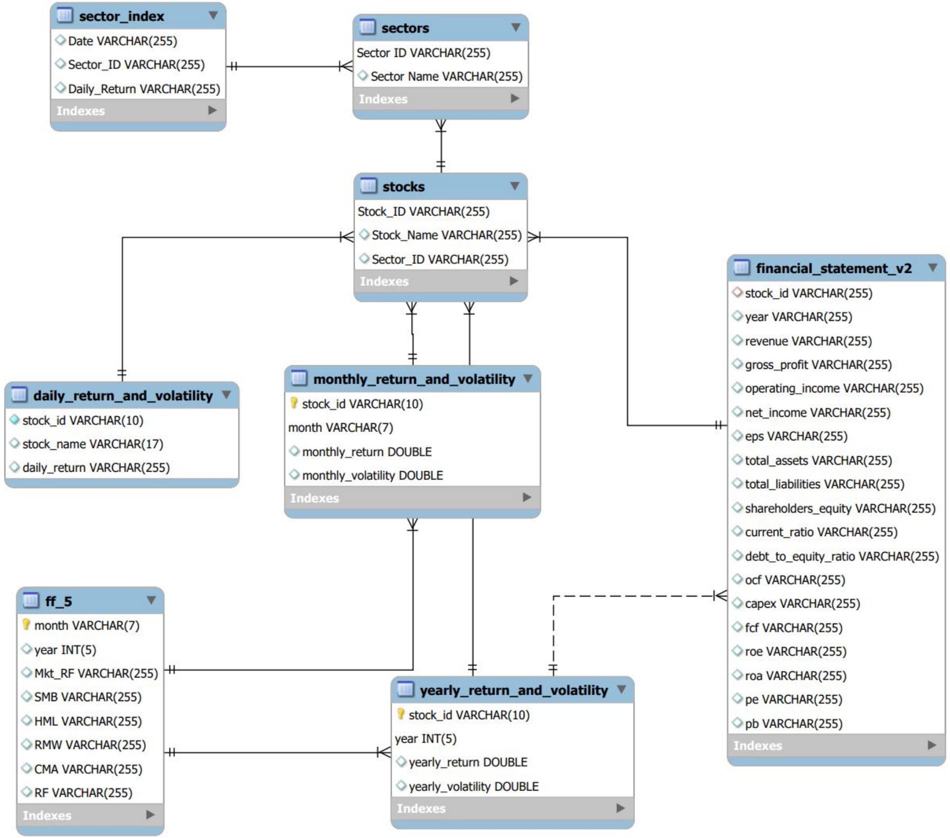
2nd Normalization:

- The tables satisfy 1NF.
- All non-primary-key attributes in each table are fully dependent on the primary key
- E.g: Composite Key *(stock_id, month)*

3rd Normalization:

- No transitive dependencies exist
- E.g: *sector* and *stock* Table, Foreign key *Sector_id* in *stocks*

EER Diagram



sector_index
Date VARCHAR(255)
Sector_ID VARCHAR(255)
Daily_Return VARCHAR(255)
Indexes
sectors
Sector ID VARCHAR(255)
Sector Name VARCHAR(255)
Indexes
stocks
Stock_ID VARCHAR(255)
Stock_Name VARCHAR(255)
Sector_ID VARCHAR(255)
Indexes
financial_statement_v2
stock_id VARCHAR(255)
year VARCHAR(255)
revenue VARCHAR(255)
gross_profit VARCHAR(255)
operating_income VARCHAR(255)
net_income VARCHAR(255)
eps VARCHAR(255)
total_assets VARCHAR(255)
total_liabilities VARCHAR(255)
shareholders_equity VARCHAR(255)
current_ratio VARCHAR(255)
debt_to_equity_ratio VARCHAR(255)
ocf VARCHAR(255)
capex VARCHAR(255)
fcf VARCHAR(255)
roe VARCHAR(255)
roa VARCHAR(255)
pe VARCHAR(255)
pb VARCHAR(255)
Indexes
daily_return_and_volatility
stock_id VARCHAR(10)
stock_name VARCHAR(17)
daily_return VARCHAR(255)
monthly_return_and_volatility
stock_id VARCHAR(10)
month VARCHAR(7)
monthly_return DOUBLE
monthly_volatility DOUBLE
Indexes
ff_5
month VARCHAR(7)
year INT(5)
Mkt_RF VARCHAR(255)
SMB VARCHAR(255)
HML VARCHAR(255)
RMW VARCHAR(255)
CMA VARCHAR(255)
RF VARCHAR(255)
Indexes
yearly_return_and_volatility
stock_id VARCHAR(10)
year INT(5)
yearly_return DOUBLE
yearly_volatility DOUBLE
Indexes

Tables Setup

Create new table

```
CREATE TABLE bus211a.bac AS
SELECT
    stock_id,
    date,
    'Bank of America' AS stock_Name,
    `Return` AS `daily_return`
FROM bus211a.bac_origin;
```

Clean, delete empty dates

```
DELETE FROM bac WHERE date IS NULL;
```

Tables Setup

Primary Keys and Foreign Keys Setup

```sql
-- Set primary key and foreign key
ALTER TABLE bac ADD PRIMARY KEY (stock_id, date);
ALTER TABLE jnj ADD PRIMARY KEY (stock_id, date);
ALTER TABLE msft ADD PRIMARY KEY (stock_id, date);
ALTER TABLE wmt ADD PRIMARY KEY (stock_id, date);
ALTER TABLE monthly_return_and_volatility ADD PRIMARY KEY (stock_id, month);
ALTER TABLE yearly_return_and_volatility ADD PRIMARY KEY (stock_id, year);
```

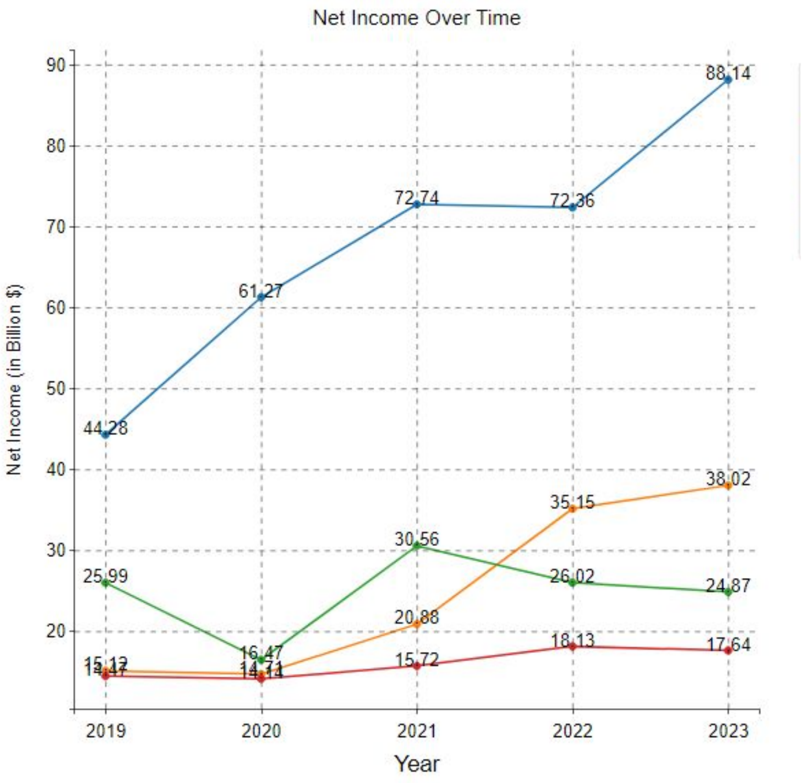
```sql
ALTER TABLE daily_return_and_volatility
ADD CONSTRAINT fk_stock_id FOREIGN KEY (stock_id) REFERENCES stocks(stock_id);
ALTER TABLE stocks
ADD CONSTRAINT fk_stock_id_daily FOREIGN KEY (stock_id) REFERENCES stocks(stock_id);
```

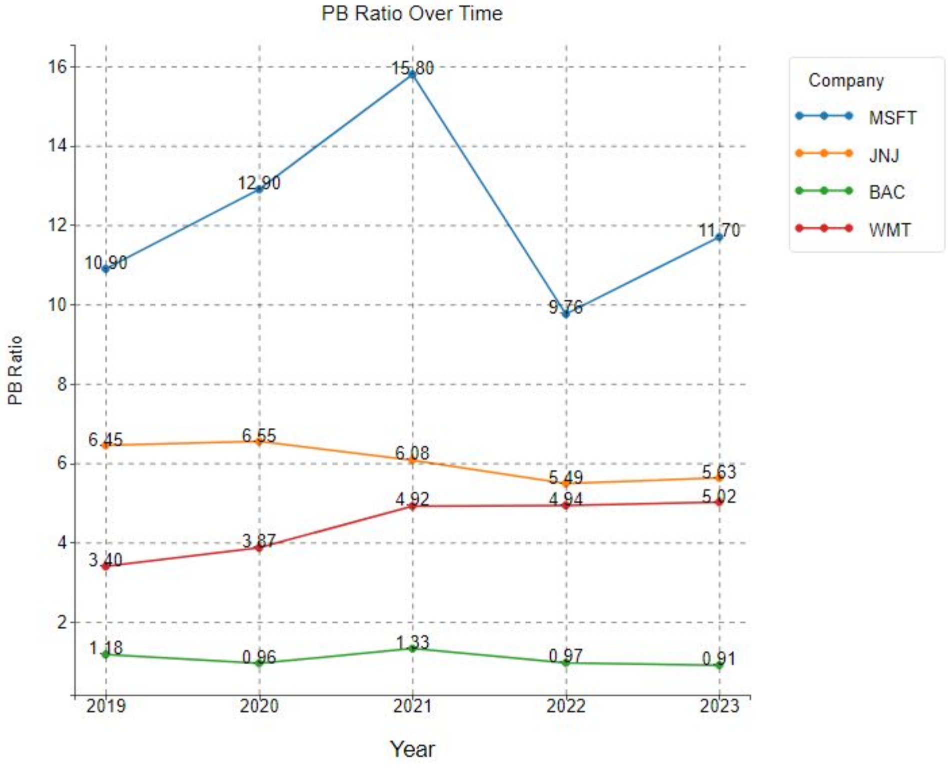
Example: Monthly Return and Volatility

```sql
-- BAC
CREATE TABLE monthly_return_and_volatility AS
SELECT
    stock_id,
    DATE_FORMAT(date, '%Y-%m') AS month,
    EXP(SUM(LOG(1 + IFNULL(daily_return, 0)))) - 1 AS monthly_return,
    STDDEV(IFNULL(daily_return, 0)) * SQRT(21) AS monthly_volatility
FROM
    bus211a.bac
GROUP BY
    stock_id, month;
```

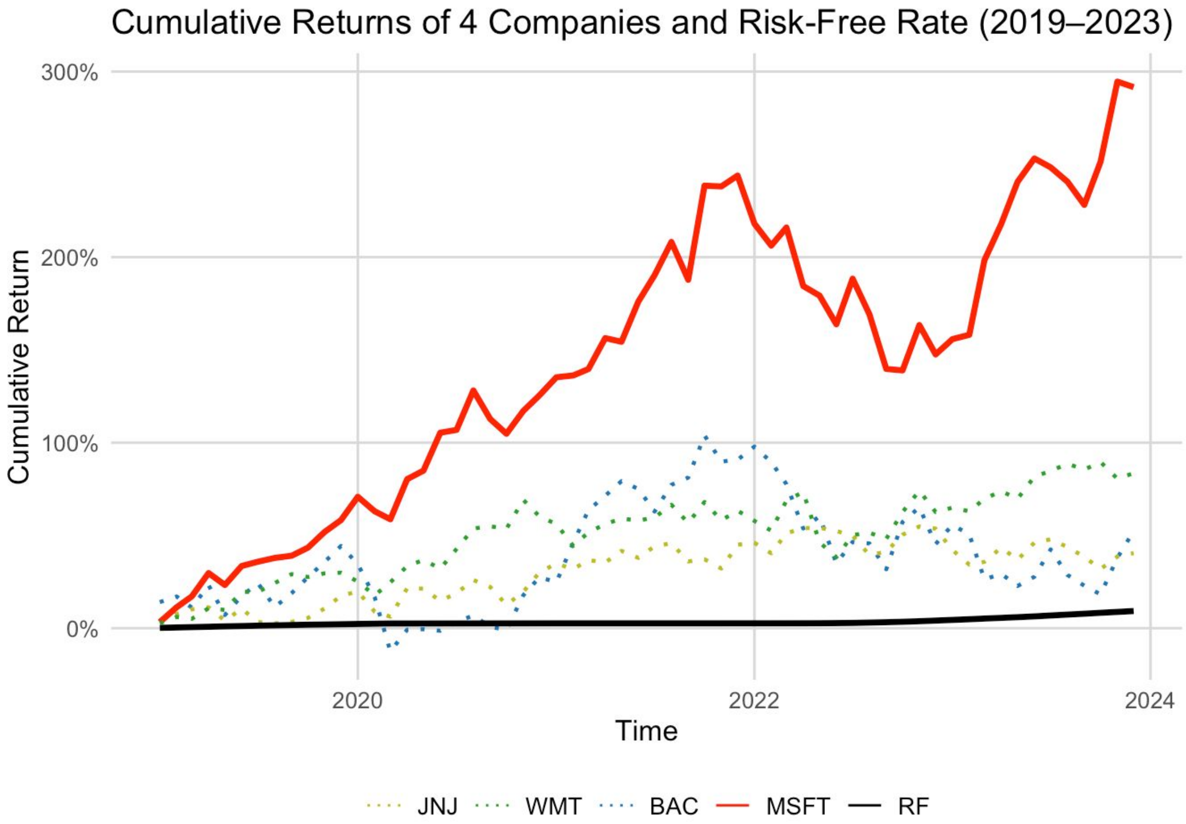
stock_id	month	monthly_return	monthly_volatility
BAC	2019-01	0.14062520475337337	9042463308876678
BAC	2019-02	0.026637047247844103	4510761813959086
BAC	2019-03	-0.051236243696740846	6713108235881994
BAC	2019-04	0.10837042576425526	.0543844038541513
BAC	2019-05	-0.130148302238147	0.064261945954439
BAC	2019-06	0.09611382932962709	6660671537163125
BAC	2019-07	0.05793210388957326	4497778220419794
BAC	2019-08	-0.10332282629671397	9622303210019825
BAC	2019-09	0.06733301782684276	6627569827046335
BAC	2019-10	0.07198958263582611	6331440538645572
BAC	2019-11	0.06555955171235572	5922425376636244
BAC	2019-12	0.06279577326872476	4397685662882672
BAC	2020-01	-0.06785918578612504	5248099209464408
BAC	2020-02	-0.1318918748216259	9998634522585205
BAC	2020-03	-0.25033291601121843	8797575407368085
BAC	2020-04	0.13282890942660486	0555111264343076
BAC	2020-05	0.0029074914608369085	7682279611135407
BAC	2020-06	-0.008470859343252313	7342478145210605
BAC	2020-07	0.047581103476182385	9823702036653935
BAC	2020-08	0.03456659309129484	6836481041407473
BAC	2020-09	-0.05758356608513948	8457191709905555
BAC	2020-10	-0.016189542522978995	9092558830769938

Financial data analysis





CUMULATIVE RETURN OF 4 COMPANIES AND RF (2019 - 2023)



- Microsoft (MSFT) has significantly outperformed the other three companies, with a cumulative return exceeding 250%.
- Bank of America (BAC), Johnson & Johnson (JNJ), and Walmart (WMT) showed steady but lower cumulative returns, ranging between 40% and 100%.
- The Risk-Free Rate (RF), represented as the baseline, reflects minimal growth over the same period.

So, for a 5-year range, investing in

AT LEAST ONE OF THESE STOCKS

offers

BETTER FINANCIAL GROWTH

compared to saving money in a bank.

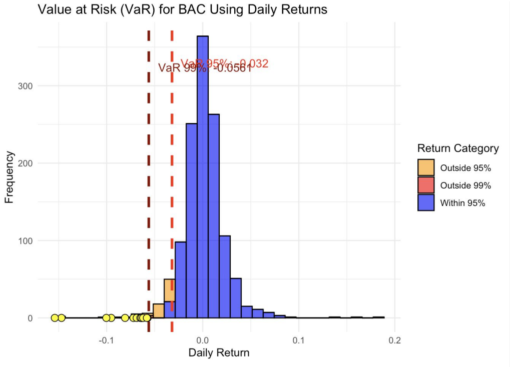
Value-At-Risk

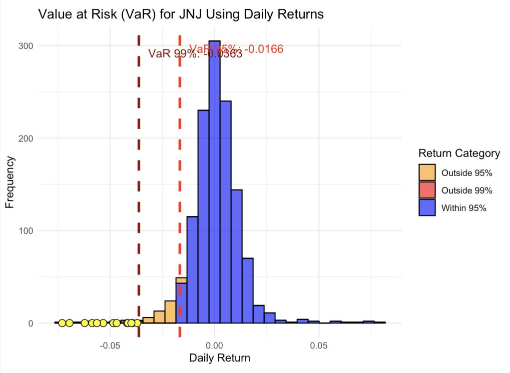
is a financial metric that **quantifies the potential loss in value of an investment or portfolio** over a specified time period, and at a **given confidence level**.

01	02	03
LOSS THRESHOLD	**CONFIDENCE LEVEL**	**TIME HORIZON**
Maximum Expected Loss	95%, 99%	2019-2023

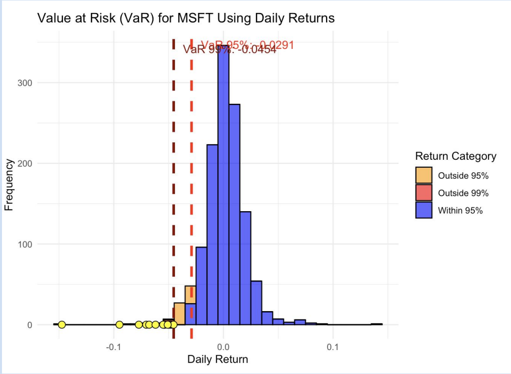
Value-At-Risk

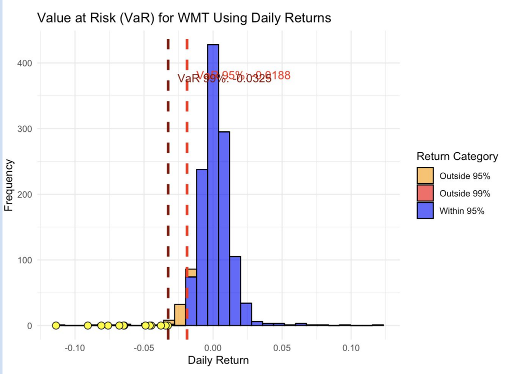
BAC & JNJ



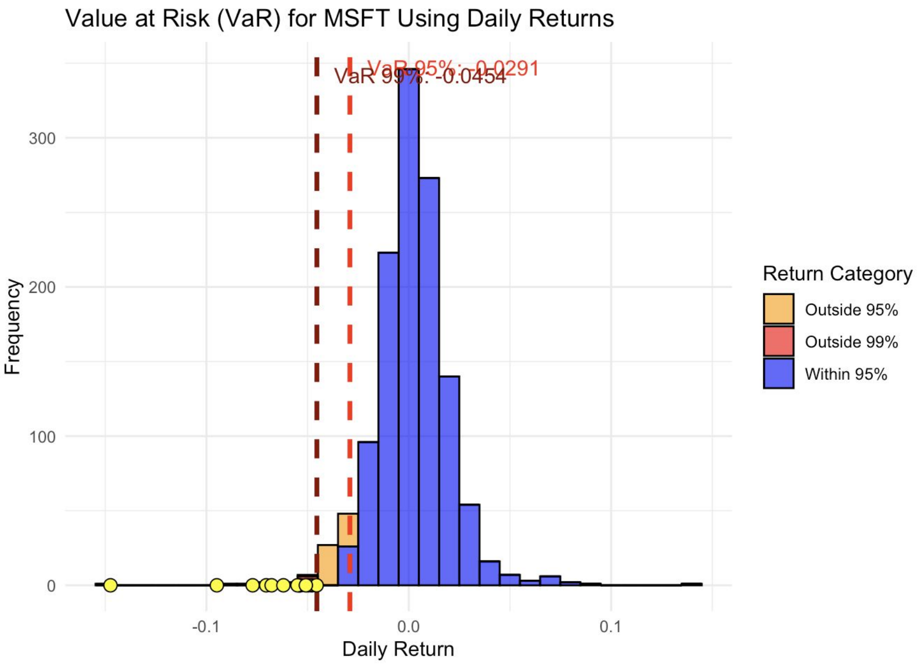


MSFT & WMT





Value-At-Risk



MSFT

```
## --- MSFT ---
## VaR 95% (Matched Return): -0.0291
## VaR 99% (Matched Return): -0.0454
```

Value-At-Risk

Company/ Confidence Level	95%	99%
MSFT	-2.91%	-4.54%
BAC	**-3.2%**	**-5.61%**
JNJ	**-1.66%**	-3.63%
WMT	**-1.88%**	**-3.25%**

Expected-Shortfall

also known as **Conditional Value at Risk (CVaR)**, is a risk measure that estimates **the average loss** in the worst-case scenarios beyond a certain Value at Risk (VaR) threshold.

Dollar-Expected-Shortfall

01

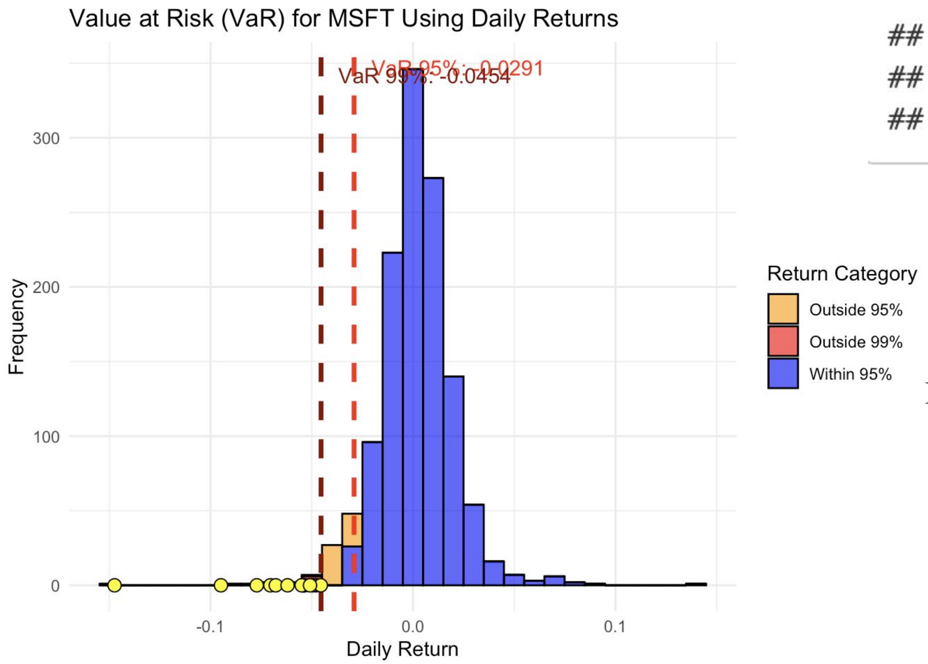
PORTFOLIO VALUE

02

AVERAGE LOSS BEYOND THE THRESHOLD

95%, 99%

Dollar-Expected-Shortfall




```
## --- MSFT ---
## VaR 95% (Matched Return): -0.0291
## VaR 99% (Matched Return): -0.0454
```

01

PORTFOLIO VALUE

02

AVERAGE LOSS BEYOND THE THRESHOLD

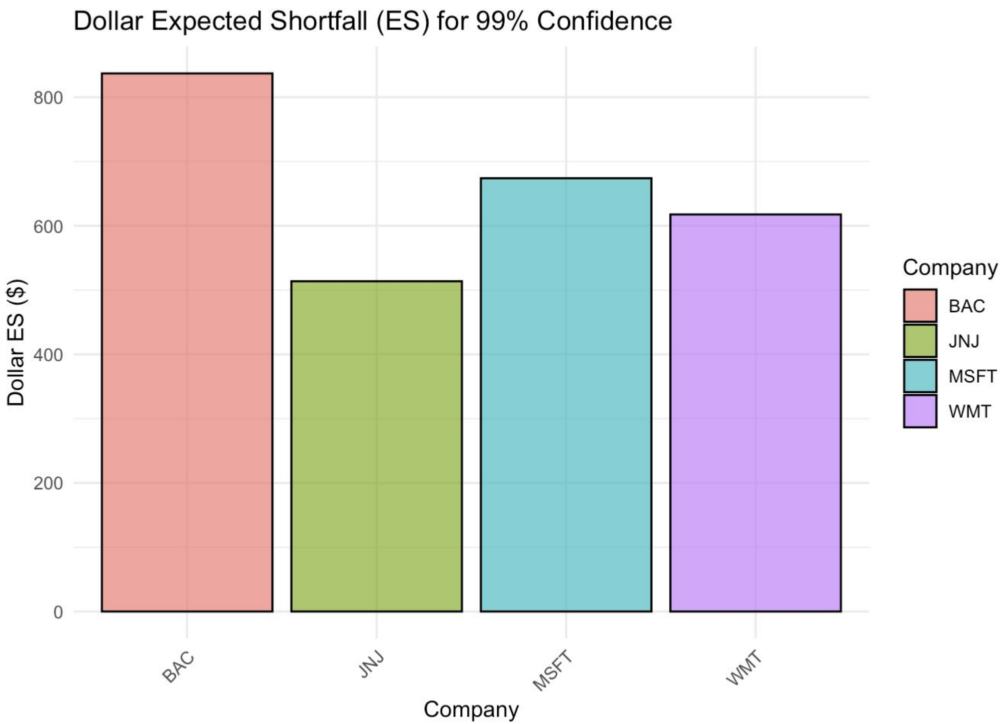
$10000 * (-4.24%) =

-$424.81

Dollar-Expected-Shortfall


```
## --- BAC ---
## VaR 95% (Potential Loss): $ 319.83
## VaR 99% (Potential Loss): $ 561.46
## ES 95% (Average Loss Beyond VaR): $ 494.59
## ES 99% (Average Loss Beyond VaR): $ 837.05
##
## --- JNJ ---
## VaR 95% (Potential Loss): $ 166.24
## VaR 99% (Potential Loss): $ 362.56
## ES 95% (Average Loss Beyond VaR): $ 291.51
## ES 99% (Average Loss Beyond VaR): $ 513.75
##
## --- MSFT ---
## VaR 95% (Potential Loss): $ 290.83
## VaR 99% (Potential Loss): $ 454.02
## ES 95% (Average Loss Beyond VaR): $ 424.81
## ES 99% (Average Loss Beyond VaR): $ 674
##
## --- WMT ---
## VaR 95% (Potential Loss): $ 188.41
## VaR 99% (Potential Loss): $ 325.23
## ES 95% (Average Loss Beyond VaR): $ 309.86
## ES 99% (Average Loss Beyond VaR): $ 617.64
```

Dollar-Expected-Shortfall




```
## --- BAC ---
## VaR 95% (Potential Loss): $ 319.83
## VaR 99% (Potential Loss): $ 561.46
## ES 95% (Average Loss Beyond VaR): $ 494.59
## ES 99% (Average Loss Beyond VaR): $ 837.05
##
## --- JNJ ---
## VaR 95% (Potential Loss): $ 166.24
## VaR 99% (Potential Loss): $ 362.56
## ES 95% (Average Loss Beyond VaR): $ 291.51
## ES 99% (Average Loss Beyond VaR): $ 513.75
##
## --- MSFT ---
## VaR 95% (Potential Loss): $ 290.83
## VaR 99% (Potential Loss): $ 454.02
## ES 95% (Average Loss Beyond VaR): $ 424.81
## ES 99% (Average Loss Beyond VaR): $ 674
##
## --- WMT ---
## VaR 95% (Potential Loss): $ 188.41
## VaR 99% (Potential Loss): $ 325.23
## ES 95% (Average Loss Beyond VaR): $ 309.86
## ES 99% (Average Loss Beyond VaR): $ 617.64
```

Market Sensitivity and Alpha

CAPM regression model: $Ri - Rf = \alpha + \beta(Rm - Rf) + \epsilon$

Alpha: Shows a stock's excess return beyond market influence.

Beta: Measures a stock's reaction to market changes.

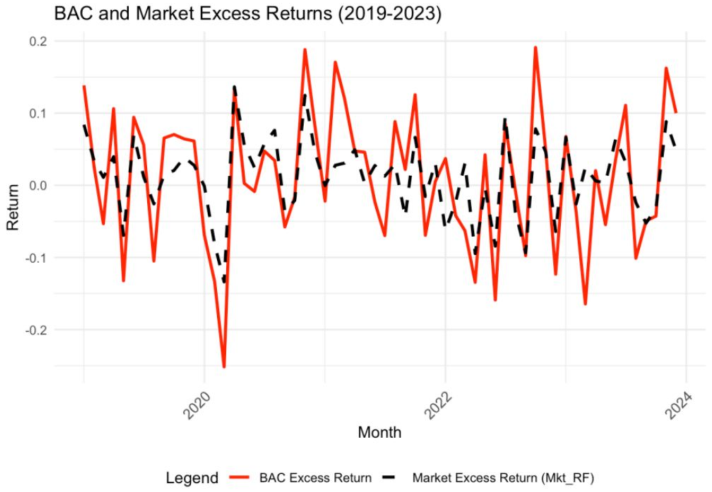
```
##
## --- Regression Results for Stock: BAC ---
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.192782 -0.040665  0.000118  0.041577  0.138530
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.006135   0.007741  -0.792    0.431
## X            1.376238   0.137269  10.026 2.82e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

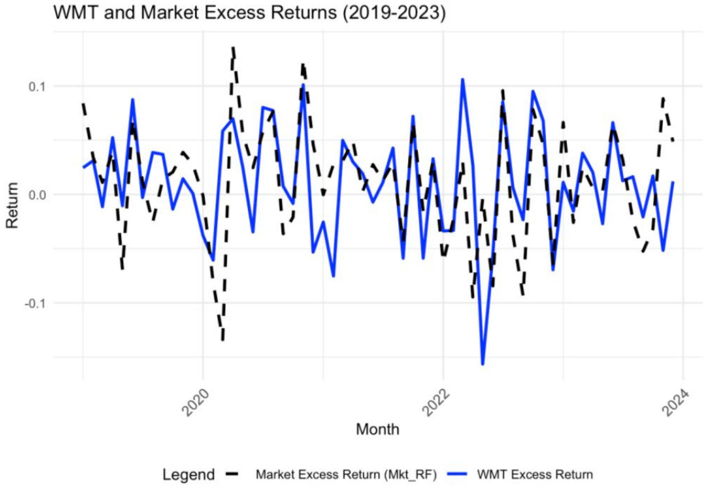
```
## --- Regression Results for Stock: JNJ ---
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.110777 -0.029752  0.002154  0.029014  0.082125
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.000467   0.005510  -0.085    0.933
## X            0.494727   0.097707   5.063 4.46e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## --- Regression Results for Stock: MSFT ---
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.083869 -0.033906 -0.004688  0.027679  0.117881
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.013701   0.005838   2.347   0.0224 *
## X           0.824570   0.103535   7.964 7.03e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## --- Regression Results for Stock: WMT ---
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15972 -0.02324  0.00197  0.02729  0.11303
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.004753   0.005988   0.794 0.430591
## X           0.442539   0.106185   4.168 0.000104 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Market Sensitivity and Alpha





BAC (High Beta, 1.38): Reacts strongly to market changes, with large fluctuations.

WMT (Low Beta, 0.44): Stable and defensive, with minimal reaction to market movements.

Deeper Factor Analysis

Mkt_RF: Stock sensitivity to market risk

(Positive; Significant)

SMB: Size effect; small-cap vs. large-cap sensitivity

(Not significant)

HML: Value vs. growth factor

(Negative; Significant)

RMW: Profitability effect

(Not significant)

CMA: Investment strategy

(Positive; Significant)

```
## Call:
## lm(formula = excess_return ~ Mkt_RF + SMB + HML + RMW + CMA,
##     data = wmt_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14802 -0.01553 -0.00024  0.02894  0.06257
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0008123  0.0057387   0.142    0.888
## Mkt_RF       0.5619282  0.1139994   4.929 8.23e-06 ***
## SMB         -0.2533236  0.2485259  -1.019    0.313
## HML         -0.3867582  0.1847434  -2.093    0.041 *
## RMW          0.2551931  0.2641824   0.966    0.338
## CMA          0.5293996  0.2628644   2.014    0.049 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04203 on 54 degrees of freedom
## Multiple R-squared:  0.3845, Adjusted R-squared:  0.3276
## F-statistic: 6.748 on 5 and 54 DF,  p-value: 5.961e-05
```

Sharpe Ratio Analysis and Key Comparisons

Comprehensive Comparison Table

Metric	BAC	MSFT	WMT	JNJ
Financial Statements	Moderate	Strong	Moderate	Strong
Sharpe Ratio	0.4644	1.2525	0.7355	0.4508
Sharpe Ratio Level	Low	**High**	Moderate	Low
Beta	1.38	0.82	0.44	0.49
Alpha	Insignificant	**Significant**	Insignificant	Insignificant
VaR (99%)	$561.46	$454.02	$325.23	$362.56
ES (99%)	$837.05	$674.00	$617.64	$513.75
Cumulative Return	High	**Highest**	Moderate	Low

MSFT: Strong Buy – Highest Sharpe Ratio, significant alpha, and best overall returns for growth-oriented investors

WMT: Buy – Moderate Sharpe Ratio, defensive traits, and low risk, ideal for stability-focused investors

JNJ: Hold – Low Sharpe Ratio with solid defensiveness, but limited growth potential compared to WMT

BAC: Sell – Low Sharpe Ratio and high beta signal inconsistent performance and higher risk

References

- https://corporatefinanceinstitute.com/resources/equities/2010-flash-crash/
- https://www.marketswiki.com/wiki/Market_Information_Data_Analytics_System
- https://www.sec.gov/securities-topics/market-structure-analytics/midas-market-information-data-analytics-system#:~:text=SEC.gov%20%7C%20MIDAS%3A%20Market%20Information%20Data%20Analytics%20System
- https://www.forbes.com/advisor/investing/modern-portfolio-theory/
- https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/
- https://www.sciencedirect.com/science/article/pii/S0957417423008485
- https://www.simplilearn.com/data-visualization-tools-article#what_do_the_best_data_visualization_tools_have_in_common
- https://www.ibm.com/topics/data-visualization?utm_content=SRCWW&p1=Search&p4=43700074739286868&p5=p&p9=58700008227896143&gclid=Cj0KCQjwgL-3BhDnARIsAL6KZ6-Amw2tXOXhdKaPr_e3SkgQ2Yi4hGNQ5QTFFdGQGNjQhQX6-NGo2qwaAnW7EALw_wcB&gclsrc=aw.ds