Measurement of portfolio efficiency

prof. dr. Audrius Kabašinskas

Green Digital Finance - Training Week

KTU, 30 June – 04 July 2025

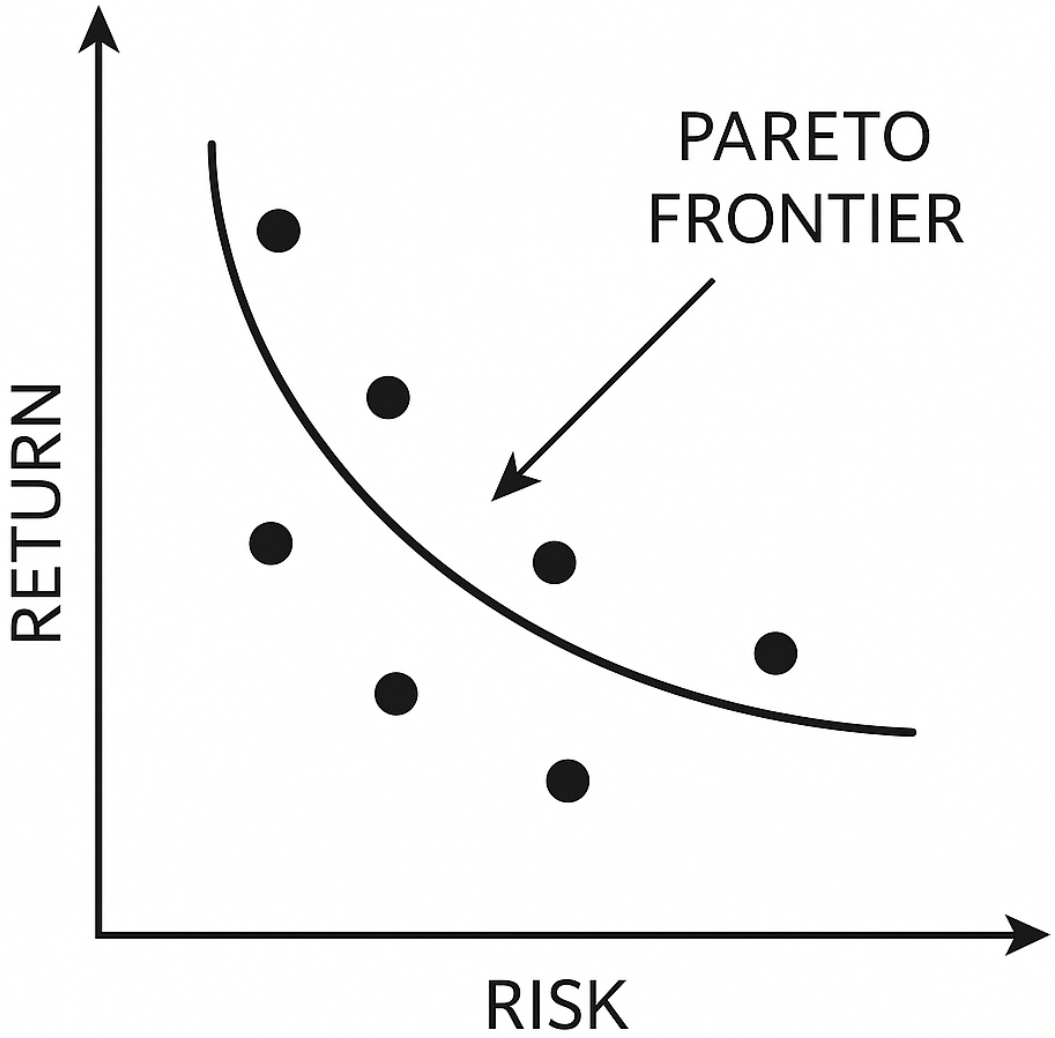
Why Measure Investment Efficiency?

- Assessing risk-adjusted return of investments.
- Comparing different investment strategies or portfolios.
- Identifying under/over-performing assets.
- Supporting decision making in portfolio management.
- Regulatory and client reporting requirements.

Pareto Frontier in Finance

- **Pareto Frontier**: Set of optimal portfolios offering maximum return for a given level of risk.
- Efficient frontier under Modern Portfolio Theory.
- Portfolios on the frontier are **not dominated** by others.

Pareto Frontier in Finance



Outline

Overview

Fundamental categories of performance analysis:

- Return Measures
- Risk Measures
- Risk-adjusted Return Measures
- Drawdown Measures
- Distribution and Tail Risk Measures

Return Measures

- **Arithmetic Return**: Average of periodic returns.
- **Geometric Return**: Compound return over time.
- **Annualized Return**: Scaled to a yearly period.
- Essential for understanding the reward aspect of investment performance.
- excess returns
- etc.
- **Further reading:** Sharpe (1993) - Performance Measurement.

Risk Measures

- **Volatility**: Standard deviation of returns.
- **Downside Risk**: Focuses on negative deviations.
- **Value-at-Risk (VaR)**: Expected maximum loss at a given confidence level.
- **Expected Shortfall (CVaR)**: Average loss beyond VaR.
- etc.

Risk Measures

Further reading: Rockafellar and Uryasev (2002) - Conditional Value-at-Risk.

Risk-adjusted Return Measures

- **Sharpe Ratio**: Excess return per unit of risk.
- **Sortino Ratio**: Adjusted for downside deviation.
- **Treynor Ratio**: Uses beta instead of volatility.
- **Information Ratio**: Return relative to a benchmark.
- etc.
- **Further reading:** Sortino and Price (1991) - Performance Measurement in a Downside Risk Framework.

Drawdown Measures

- **Maximum Drawdown**: Largest peak-to-trough decline.
- **Average Drawdown**: Mean of all drawdowns.
- **Recovery Time**: Time to regain peak value.
- Helps quantify capital preservation under stress.
- etc.
- **Further reading:** Chekhlov et al. (2005) - Drawdown: From Practice to Theory and Back Again.

Distribution and Tail Risk Measures

- **Rachev Ratio**: average of best returns / average of worst loses

- Stochastic dominance
- etc.
- **Further reading:** wiki.

Benchmark vs Non-Benchmark Measures

- **Benchmark-based Measures** evaluate performance relative to a market index or peer group.
  - Examples: Information Ratio, Jensen's Alpha, Treynor Ratio.
  - Useful for active portfolio managers.
- **Non-Benchmark Measures** assess absolute performance without reference.
  - Examples: Sharpe Ratio, Sortino Ratio, Maximum Drawdown.
  - Useful for absolute return strategies.
- **Key Differences:**
  - Benchmark measures are context-sensitive.
  - Non-benchmark metrics focus purely on standalone risk/return.
- **Further reading:** Grinold and Kahn (2000) - Active Portfolio Management.

Outline

Approach

In this part above 50 top performance measures are presented.

- Derived from `PerformanceAnalytics` R package.
- python has no nice and unified package
- Widely used in academia and practice.
- Presented with formula, interpretation, R function and recommended python packages.

Non-benchmark and benchmark-based measurements are separated.

1. Sharpe Ratio (Part 1)

- **Formula:** $S = \frac{R_p - R_f}{\sigma_p}$
- **Inventor:** William F. Sharpe, 1966 (revised in 1994).
- **Purpose:** Quantifies excess return per unit of total risk (volatility).
- **Usage:** Widely used in:
  - Portfolio management
  - Mutual fund evaluation
  - Hedge fund performance
- **Data Requirements:**
  - Periodic portfolio returns
  - Corresponding risk-free rate
  - Standard deviation of returns

1. Sharpe Ratio (Part 2)

- **Strengths:**
  - Easy to interpret and compare across assets
  - Applicable to various asset classes
  - Encourages diversification to reduce volatility
- **Limitations:**
  - Assumes normally distributed returns
  - Penalizes both upward and downward volatility
- **R function:** `SharpeRatio.annualized()`
- **Further reading:** Sharpe (1994) - The Sharpe Ratio

1a. Variants of Sharpe Ratio

Adjusted Sharpe Ratios based on Tail Risk

- **VaR-based Sharpe Ratio:**
  - $S_{VaR} = \frac{R_p - R_f}{VaR}$
  - Accounts for potential extreme losses.
  - **R:** `SharpeRatio` with `FUN = "VaR"`
- **ES-based Sharpe Ratio:**
  - $S_{ES} = \frac{R_p - R_f}{ES}$
  - Uses Conditional Value-at-Risk for tail risk.
  - **R:** `SharpeRatio` with `FUN = "ES"`

1b. Sharpe Ratio in Python

Implementation in Python (with NumPy and pandas)

- **Basic Sharpe Ratio:** `(portfolio_returns.mean() - risk_free_rate) / portfolio_returns.std()`
- **Annualized Sharpe Ratio:** `sharpe_annual = sharpe_ratio * np.sqrt(252)`
- **With Tail Risk (e.g. ES):** Use libraries like `QuantStats`, `riskfolio-lib`

2. Sortino Ratio (Part 1)

- **Formula:** $S_{Sortino} = \frac{R_p - R_f}{\sigma_d}$
- **Explanation:** R_p is portfolio return, R_f is the risk-free rate, σ_d is the standard deviation of negative returns (downside deviation).
- **Downside deviation:** $\sigma_d = \sqrt{\frac{1}{n}\sum_{t=1}^{n} \min(R_t - T, 0)^2}$, where T is the target or minimum acceptable return.
- **Inventor:** Frank A. Sortino
- **Purpose:** Focuses only on downside risk (ignores upside volatility)
- **Usage:** Preferred when return distributions are skewed

2. Sortino Ratio (Part 2)

- **Data Requirements:** Returns, risk-free rate, downside deviation
- **Strengths:**
  - More realistic in non-normal return scenarios
  - Penalizes only harmful volatility
- **Limitations:** Less common in traditional reporting
- **R function:** `SortinoRatio()`
- **Python:** `riskfolio-lib`, `QuantStats`
- **Further reading:** Sortino and Price (1991)

3. Calmar Ratio (Part 1)

- **Formula:** $C = \frac{R_a}{\text{Max Drawdown}}$
- **Explanation:** R_a is the annualized return, and Max Drawdown is the largest peak-to-trough decline in portfolio value.
- **Inventor:** Created by Terry W. Young in the 1990s.
- **Purpose:** Measures risk-adjusted return using drawdown as the risk metric.
- **Usage:** Popular in hedge fund and managed futures evaluation where drawdown is a key concern.
- **Data Requirements:** Time series of portfolio returns.

3. Calmar Ratio (Part 2)

- **Strengths:**
  - Focuses on capital preservation
  - Suitable for non-normal, skewed return distributions
- **Limitations:**
  - Max Drawdown is path-dependent and sensitive to outliers
  - Does not consider frequency of drawdowns
- **R function:** `CalmarRatio()`
- **Python:** `riskfolio-lib`, `QuantStats`
- **Further reading:** Young (1991) - Calmar Ratio for Managed Futures

4. Omega Ratio (Part 1)

- **Formula:** $\Omega(T) = \dfrac{\int_{T}^{\infty} [1 - F(r)]\, dr}{\int_{-\infty}^{T} F(r)\, dr}$
- **Explanation:** Ratio of gains above a threshold T to losses below T, based on the cumulative distribution function $F(r)$.
- **Inventors:** Keating and Shadwick (2002)
- **Purpose:** Generalized risk-adjusted performance measure valid for all return distributions.
- **Usage:** Popular in hedge funds and asymmetric return strategies.
- **Data Requirements:** Full return distribution and threshold return T.

4. Omega Ratio (Part 2)

- **Strengths:**
  - Accounts for skewness and kurtosis in returns
  - No assumption of normality
  - Flexible threshold selection
- **Limitations:**
  - Computationally intensive for empirical data
  - Interpretation less intuitive than Sharpe or Sortino
- **R function:** `Omega()` from PerformanceAnalytics
- **Python:** `riskfolio-lib`, `QuantStats`
- **Further reading:** Keating and Shadwick (2002) - A Universal Performance Measure

5. Gain-Loss Ratio (Part 1)

- **Formula:** $GL = \frac{\text{Average Gain}}{\text{Average Loss}}$
- **Explanation:** Measures the magnitude of gains relative to losses.
- **Purpose:** Indicates how well gains compensate for losses.
- **Usage:** Useful in evaluating asymmetric return strategies.
- **Data Requirements:** Full return series to separate gains and losses.

5. Gain-Loss Ratio (Part 2)

- **Strengths:**
  - Simple and intuitive interpretation
  - Works well for non-normal distributions
- **Limitations:** Does not account for volatility or timing of returns
- **R function:** `GainLossRatio()`
- **Python:** `riskfolio-lib`, `QuantStats`
- **Further reading:** Le Sourd (2007) - Performance Measures for Investment Funds

6. Sterling Ratio (Part 1)

- **Formula:** $SR = \frac{R_p - R_f}{\text{Average Drawdown}}$
- **Explanation:** Compares excess return to average drawdown.
- **Purpose:** Measures return per unit of drawdown risk.
- **Usage:** Applied in evaluating risk-sensitive investment strategies.
- **Data Requirements:** Portfolio returns and drawdown series.

6. Sterling Ratio (Part 2)

- **Strengths:**
  - Focus on capital preservation
  - Good for evaluating tactical trading strategies
- **Limitations:** Sensitive to method of drawdown calculation
- **R function:** `SterlingRatio()`
- **Python:** custom
- **Further reading:** Sterling (2003) - Drawdown Measures

7. Burke Ratio (Part 1)

- **Formula:** $BR = \frac{R_a}{\sqrt{\sum_{i=1}^{n} DD_i^2}}$
- **Explanation:** Ratio of annualized return to the square root of the sum of squared drawdowns.
- **Purpose:** Penalizes more frequent and deeper drawdowns.
- **Usage:** Used to evaluate performance accounting for drawdown magnitude.
- **Data Requirements:** Return data and complete drawdown series.

7. Burke Ratio (Part 2)

- **Strengths:**
  - Penalizes deeper and more frequent drawdowns
  - Useful for downside-focused strategies
- **Limitations:** Less intuitive than Max Drawdown-based ratios
- **R function:** `BurkeRatio()`
- **Python:** custom
- **Further reading:** Le Sourd (2007)

8. Martin Ratio (Ulcer Performance Index) (Part 1)

- **Formula:** $MR = \frac{R_a}{UI}$
- **Ulcer Index:** $UI = \sqrt{\frac{1}{n}\sum_{i=1}^{n} D_i^2}$, where D_i is the percentage drawdown on day i
- **Explanation:** Return divided by Ulcer Index, which measures the depth and duration of drawdowns.
- **Purpose:** Focuses on downside volatility.
- **Usage:** Common in conservative and retirement portfolios.
- **Data Requirements:** Return series and drawdown durations.

8. Martin Ratio (Ulcer Performance Index) (Part 2)

- **Strengths:**
  - Reflects both depth and persistence of drawdowns
  - More comprehensive than Max Drawdown alone
- **Limitations:** Requires longer return history for reliable results
- **R function:** `MartinRatio()`
- **Python:** custom
- **Further reading:** Martin (1987) - Ulcer Index

9. Pain Ratio (Part 1)

- **Formula:** $PR = \frac{R_a}{\text{Pain Index}}$
- **Pain Index:** $PI = \frac{1}{T} \sum_{t=1}^{T} |DD_t|$, where DD_t is the drawdown at time t
- **Explanation:** Measures return relative to the average drawdown.
- **Purpose:** Like the Calmar Ratio but based on average drawdown rather than maximum.
- **Usage:** Hedge funds and alternative strategies.
- **Data Requirements:** Return and drawdown series.

9. Pain Ratio (Part 2)

- **Strengths:**
  - Captures ongoing drawdown pain
  - Penalizes frequent shallow losses
- **Limitations:** Interpretation can be less intuitive
- **R function:** `PainRatio()`
- **Python:** custom
- **Further reading:** Brown et al. (2014)

10. Tail Ratio (Part 1)

- **Formula:** $TR = \frac{\text{95th percentile return}}{\text{5th percentile return}}$
- **Explanation:** Compares upside tail to downside tail.
- **Purpose:** Measures tail asymmetry of the return distribution.
- **Usage:** Risk profiling and evaluation of fat-tailed strategies.
- **Data Requirements:** Full return distribution.

10. Tail Ratio (Part 2)

- **Strengths:**
  - Simple and distribution-free
  - Highlights extreme risk vs reward
- **Limitations:** Sensitive to outliers
- **R function:** custom
- **Python:** `QuantStats`
- **Further reading:** Bailey and López de Prado (2013)

11. Value at Risk (VaR) (Part 1)

- **Formula:** $VaR_\alpha = \text{Quantile}_\alpha(R)$
- **Explanation:** Measures the worst expected loss over a given time at a specific confidence level.
- **Purpose:** Quantifies tail risk of the distribution.
- **Usage:** Risk compliance, portfolio risk management.
- **Data Requirements:** Historical returns and selected confidence level.

11. Value at Risk (VaR) (Part 2)

- **Strengths:**
  - Intuitive for reporting loss risk
  - Standard in financial institutions
- **Limitations:**
  - Ignores losses beyond VaR threshold
  - Non-subadditive in some forms
- **R function:** `VaR()`
- **Python:** `QuantStats`
- **Further reading:** Jorion (2000)

12. Conditional Value at Risk (CVaR / Expected Shortfall) (Part 1)

- **Formula:** $CVaR_\alpha = E[R|R \leq VaR_\alpha]$
- **Explanation:** Expected loss given that the loss exceeds the VaR level.
- **Purpose:** Captures tail risk better than VaR.
- **Usage:** Advanced risk analytics and portfolio optimization.
- **Data Requirements:** Return series and confidence level.

12. Conditional Value at Risk (CVaR / Expected Shortfall) (Part 2)

- **Strengths:**
  - Accounts for extreme loss scenarios
  - Coherent risk measure (subadditive)
- **Limitations:** Requires more assumptions or simulations
- **R function:** `ES()`
- **Python:** `QuantStats`
- **Further reading:** Rockafellar and Uryasev (2002)

13. Downside Deviation (Part 1)

- **Formula:** $\sigma_d = \sqrt{\frac{1}{n}\sum_{t=1}^{n} \min(R_t - T, 0)^2}$
- **Explanation:** Measures standard deviation of returns that fall below a threshold return T.
- **Purpose:** Quantifies risk of falling short of a minimum acceptable return.
- **Usage:** Used in risk-adjusted return measures like Sortino Ratio.
- **Data Requirements:** Return series and threshold return.

13. Downside Deviation (Part 2)

- **Strengths:**
  - Focuses only on downside risk
  - More meaningful in asymmetric return distributions
- **Limitations:** Requires specification of threshold
- **R function:** `DownsideDeviation()`
- **Python:** `riskfolio-lib` (special case of LPM)
- **Further reading:** Sortino and Price (1991)

14. Drawdown Deviation (Part 1)

- **Formula:** $\sigma_{DD} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(DD_t - \bar{DD})^2}$
- **Explanation:** Standard deviation of drawdown values over time.
- **Purpose:** Measures volatility of drawdowns.
- **Usage:** Risk control in high-drawdown volatility strategies.
- **Data Requirements:** Drawdown series.

14. Drawdown Deviation (Part 2)

- **Strengths:**
  - Captures variability of drawdown exposure
  - Helps evaluate consistency of capital preservation
- **Limitations:** Not standardized in reporting
- **R function:** `StdDevDrawdown()` from PerformanceAnalytics
- **Python:** custom
- **Further reading:** Le Sourd (2007)

15. Rachev Ratio (Part 1)

- **Formula:** $RR = \frac{CVaR_{1-\alpha}(R)}{|CVaR_{\alpha}(R)|}$
- **Explanation:** Compares the expected gain in the right tail to the expected loss in the left tail.
- **Purpose:** Evaluates the asymmetry of risk and reward in the tails of the distribution.
- **Usage:** Risk-sensitive and tail-hedging strategies.
- **Data Requirements:** Return series and selected confidence level α.

15. Rachev Ratio (Part 2)

- **Strengths:**
  - Directly captures tail asymmetry
  - Useful for hedge funds and asymmetric strategies
- **Limitations:**
  - Requires estimation of two conditional tail distributions
  - Interpretation depends on selected confidence level
- **R function:** `RachevRatio()`
- **Python:** custom
- **Further reading:** M. Rachev et al. (2013) - Fat-Tailed and Skewed Asset Return Distributions

16. Adjusted Sharpe Ratio (Part 1)

- **Formula:** $ASR = S \times \left[1 + \frac{S_k}{6} \times S - \frac{(K_k - 3)}{24} \times S^2\right]$
- **Explanation:** Adjusts the Sharpe Ratio for non-normal return distributions using skewness (S_k) and kurtosis (K_k).
- **Purpose:** Provides a more accurate risk-adjusted return measure for skewed or fat-tailed distributions.
- **Usage:** Applied in portfolios with non-normal or alternative investments.
- **Data Requirements:** Returns, standard deviation, skewness, and kurtosis.

16. Adjusted Sharpe Ratio (Part 2)

- **Strengths:**
  - Accounts for higher moments (skewness, kurtosis)
  - More accurate than Sharpe Ratio in non-normal conditions
- **Limitations:**
  - Sensitive to estimation errors in skewness and kurtosis
  - Requires more data and statistical assumptions
- **R function:** `AdjustedSharpeRatio()`
- **Python:** custom
- **Further reading:** Pezier and White (2006)

17. Skewness (Part 1)

- **Formula:** $Skewness = \frac{\frac{1}{n}\sum(R_t - \bar{R})^3}{\sigma^3}$
- **Explanation:** Measures the asymmetry of the return distribution around its mean.
- **Purpose:** Identifies bias toward upside or downside volatility.
- **Usage:** Tail risk and portfolio risk diagnostics.
- **Data Requirements:** Return series.

17. Skewness (Part 2)

- **Strengths:** Helps interpret directional tail risk
- **Limitations:** Easily distorted by outliers
- **R function:** `skewness()` from `moments` or `e1071`
- **Python:** `quantstats`
- **Further reading:** Harvey and Siddique (2000)

18. Kurtosis (Part 1)

- **Formula:** $Kurtosis = \frac{\frac{1}{n}\sum(R_t - \bar{R})^4}{\sigma^4}$
- **Explanation:** Measures the "tailedness" of the return distribution.
- **Purpose:** Identifies presence of extreme returns (fat tails).
- **Usage:** Tail risk assessment, VaR and CVaR improvement.
- **Data Requirements:** Return series.

18. Kurtosis (Part 2)

- **Strengths:** Quantifies likelihood of outliers
- **Limitations:** Affected by small sample sizes
- **R function:** `kurtosis()` from `moments` or `e1071`
- **Python:** `quantstats`
- **Further reading:** Harvey and Siddique (2000)

19. Sortino-to-Sharpe Ratio

- **Formula:** $Ratio = \frac{Sortino}{Sharpe}$
- **Explanation:** Compares downside-risk focus (Sortino) to total-risk focus (Sharpe).
- **Purpose:** Indicates the degree of penalization from total vs downside risk.
- **Usage:** Strategy evaluation with asymmetrical return distributions.
- **R function:** Custom
- **Python:** Custom via riskfolio-lib
- **Further reading:** Sortino and Price (1991)

20. Upside Potential Ratio (Part 1)

- **Formula:** $UPR = \frac{\text{Average upside deviation above target}}{\text{Downside deviation}}$
- **Explanation:** Ratio of upside volatility (returns above target) to downside volatility.
- **Purpose:** Measures how much positive performance compensates for negative deviation.
- **Usage:** Suitable for funds where upside targeting is strategic (e.g., structured products).
- **Data Requirements:** Return series and target return (MAR).

20. Upside Potential Ratio (Part 2)

- **Strengths:**
  - Rewards portfolios with frequent or strong upside surprises
  - Good for evaluating non-normal or skewed distributions
- **Limitations:**
  - Requires setting a subjective target return
  - Less known than Sortino or Sharpe
- **R function:** `UpsidePotentialRatio()`
- **Python:** Custom
- **Further reading:** Sortino and van der Meer (2000)

21. Upside Risk (Part 1)

- **Formula:** $UR = \sqrt{\frac{1}{n}\sum_{t=1}^{n} \max(R_t - T, 0)^2}$
- **Explanation:** Measures the volatility of returns that exceed a target threshold.
- **Purpose:** Quantifies opportunity-related volatility rather than downside loss.
- **Usage:** Used to assess the consistency and magnitude of favorable returns.
- **Data Requirements:** Time series of returns and a target (e.g., risk-free rate or MAR).

21. Upside Risk (Part 2)

- **Strengths:**
  - Highlights positive deviations above target
  - Complements downside risk measures for more complete analysis
- **Limitations:**
  - Not penalized in conventional risk-adjusted ratios
  - Requires consistent target definition
- **R function:** `UpsideRisk()`
- **Python:** Custom
- **Further reading:** Sortino and van der Meer (2000)

22. Bernardo-Ledoit Ratio (Part 1)

- **Formula:** $BL = \frac{\mathbb{E}[R|R>0]}{|\mathbb{E}[R|R<0]|}$
- **Explanation:** Ratio of expected gains to expected losses.
- **Purpose:** Generalizes risk-neutral preference under uncertainty.
- **Usage:** Asset pricing and portfolio selection under minimal utility assumptions.
- **Data Requirements:** Full return distribution.

22. Bernardo-Ledoit Ratio (Part 2)

- **Strengths:**
  - Interpretable under behavioral and risk-averse frameworks
  - Equivalent to Gain-Loss Ratio under simple assumptions
- **Limitations:**
  - Requires separation of return distributions
  - Sensitive to tail behavior
- **R function:** `BernardoLedoitRatio`
- **Python:** `quantstats` (similar to win_loss_ratio)
- **Further reading:** Bernardo and Ledoit (2000)

23. D-Ratio (Part 1)

- **Formula:** $D = \frac{\text{Downside Deviation}}{\text{Upside Deviation}}$
- **Explanation:** Compares downside risk to total volatility.
- **Purpose:** Identifies how much of total risk is actually harmful (below a target return).
- **Usage:** Evaluates return distributions where upside and downside risks differ significantly.
- **Data Requirements:** Return series and a defined threshold (usually MAR).

23. D-Ratio (Part 2)

- **Strengths:**
  - Highlights dominance of downside vs total volatility
  - Useful in skewed or downside-sensitive strategies
  - Is similar to the Bernado Ledoit ratio but inverted and taking into account the frequency of positive and negative returns.
- **Limitations:**
  - Can be unstable with limited data
  - May overemphasize minor downside variation
- **R function:** `DRatio()`
- **Python:** custom
- **Further reading:** Le Sourd (2007)

24. Hurst Index (Part 1)

- **Formula:** Estimate based on rescaled range analysis: $R/S \propto T^H$
- **Explanation:** Quantifies the long-term memory or persistence in a return time series.
- **Purpose:** Indicates trend reinforcement ($H > 0.5$), randomness ($H = 0.5$), or mean reversion ($H < 0.5$).
- **Usage:** Detects momentum, fractality, and autocorrelation in returns.
- **Data Requirements:** Long, evenly spaced return time series.

24. Hurst Index (Part 2)

- **Strengths:**
  - Captures memory in return structures
  - Useful in forecasting and volatility modeling
- **Limitations:**
  - Requires long sample for stability
  - Not a direct performance ratio
- **R function:** `HurstIndex()`
- **Python:** `nolds` or `hurst` or `hurst_exponent`
- **Further reading:** Peters (1994), Mandelbrot and Wallis (1969)

25. Kappa Ratio (Part 1)

- **Formula:** $\text{Kappa}_n = \frac{\mathbb{E}[R-T]}{(\mathbb{LPM}_n(T))^{1/n}}$
- **Explanation:** Generalized downside risk-adjusted return metric, where n is the order of the lower partial moment.
- **Purpose:** Captures both magnitude and frequency of downside deviations.
- **Usage:** Suitable for evaluating asymmetric risk preferences and performance under non-normal return distributions.
- **Data Requirements:** Return series and target return (e.g., MAR).

25. Kappa Ratio (Part 2)

- **Strengths:**
  - Flexible framework that includes Sortino ($n = 2$) and Omega-type ratios
  - Suitable for downside-aware investors
- **Limitations:**
  - Interpretation becomes less intuitive as order n increases
  - Requires more data for higher moments
- **R function:** `Kappa()`
- **Python:** Custom
- **Further reading:** Le Sourd (2007)

26. Kelly Ratio (Part 1)

- **Formula:** $K = \frac{\mu - R_f}{\sigma^2}$
- **Explanation:** Determines the optimal fraction of capital to allocate to a risky asset to maximize long-term capital growth.
- **Purpose:** Maximizes expected logarithmic utility of wealth (geometric growth).
- **Usage:** Position sizing, leverage strategy design, algorithmic trading systems.
- **Data Requirements:** Mean return (μ), standard deviation (σ), and risk-free rate (R_f).

26. Kelly Ratio (Part 2)

- **Strengths:**
  - Theoretically optimal for long-term growth
  - Incorporates both return and risk directly
- **Limitations:**
  - Assumes known distributions and repeatable outcomes
  - Can lead to overbetting if misestimated
- **R function:** `KellyRatio()`
- **Python:** `quantstats`
- **Further reading:** MacLean, Ziemba & Blazenko (2004)

27. Prospect Ratio (Part 1)

- **Formula:** $PR = \frac{\int_{T}^{\infty}(R-T)f(R)dR}{\int_{-\infty}^{T}(T-R)^2 f(R)dR}$
- simplified:

$$ProspectRatio(R) = \frac{\frac{1}{n}\sum_{i=1}^{n}(\max(r_i,0)+2.25\cdot\min(r_i,0))-MAR}{\sigma_D}$$

- **Explanation:** Measures the trade-off between expected gains above a target and squared losses below it.
- **Purpose:** Reflects the prospect theory principle of valuing losses more than gains.
- **Usage:** Behavioral finance–oriented performance analysis.
- **Data Requirements:** Full return distribution and target return.

27. Prospect Ratio (Part 2)

- **Strengths:**
  - Penalizes downside more severely than upside rewards
  - Aligns with behavioral preferences under Prospect Theory
  - Penalizes losses quadratically, rewarding moderate and frequent gains
- **Limitations:**
  - Requires consistent definition of MAR and downside deviation
  - Requires integration over full distribution (numerical approximation)
  - Less widely adopted in traditional finance
- **R function:** ProspectRatio()
- **Python:** Custom
- **Further reading:** Le Sourd (2007), Kahneman & Tversky (1979)

Measures with Benchmark

Performance Measures that requires benchmark

1. Information Ratio (Part 1)

- **Formula:** $IR = \frac{R_p - R_b}{\sigma_{(R_p - R_b)}}$
- **Explanation:** Measures excess return over benchmark per unit of tracking error.
- **Purpose:** Evaluates consistency of active management.
- **Usage:** Comparing active portfolios against benchmarks.
- **Data Requirements:** Portfolio and benchmark return series.

1. Information Ratio (Part 2)

- **Strengths:** Suitable for active manager assessment
- **Limitations:** Sensitive to tracking error
- **R function:** `InformationRatio()`
- **Python:** `quantstats`
- **Further reading:** Goodwin (1998)

2. Alpha (Part 1)

- **Formula:** $\alpha = R_p - [R_f + \beta(R_b - R_f)]$
- **Explanation:** Measures excess return above expected return based on CAPM.
- **Purpose:** Evaluates manager skill beyond market exposure.
- **Usage:** Fundamental in asset pricing and manager evaluation.
- **Data Requirements:** Portfolio, benchmark, and risk-free rate.

2. Alpha (Part 2)

- **Strengths:** Grounded in theory; widely used
- **Limitations:** Dependent on correct beta estimation
- **R function:** `CAPM.alpha()`
- **Python:** `financetoolkit`
- **Further reading:** Jensen (1968)

3. Treynor Ratio (Part 1)

- **Formula:** $TR = \frac{R_p - R_f}{\beta}$
- **Explanation:** Measures return earned in excess of the risk-free rate per unit of systematic risk.
- **Purpose:** Suitable for diversified portfolios.
- **Usage:** Risk-adjusted performance based on market sensitivity.
- **Data Requirements:** Returns and beta.

3. Treynor Ratio (Part 2)

- **Strengths:** Works well for well-diversified portfolios
- **Limitations:** Not suitable for undiversified portfolios
- **R function:** `TreynorRatio()`
- **Python:** `financetoolkit`
- **Further reading:** Treynor (1965)

4. Modigliani-Modigliani (M^2) (Part 1)

- **Formula:** $M^2 = (R_p - R_f) \cdot \frac{\sigma_M}{\sigma_p} + R_f$
- **Explanation:** Converts Sharpe ratio into units of percentage return by adjusting for benchmark risk.
- **Purpose:** Makes performance easily comparable to a market benchmark.
- **Usage:** Institutional reporting.
- **Data Requirements:** Portfolio returns, benchmark standard deviation, and risk-free rate.

4. Modigliani-Modigliani (M^2) (Part 2)

- **Strengths:** Clear interpretation in percent return terms
- **Limitations:** Requires reliable market volatility measure
- **R function:** `Modigliani()`
- **Python:** financetoolkit
- **Further reading:** Modigliani & Modigliani (1997)

5. Tracking Error (Part 1)

- **Formula:** $TE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_{p,i} - R_{b,i})^2}$
- **Explanation:** Measures standard deviation of portfolio returns relative to benchmark.
- **Purpose:** Evaluates consistency and volatility of active management.
- **Usage:** Passive vs active strategy comparison.
- **Data Requirements:** Portfolio and benchmark returns.

5. Tracking Error (Part 2)

- **Strengths:** Emphasizes volatility of active bets
- **Limitations:** Doesn't indicate direction of deviation
- **R function:** `TrackingError()`
- **Python:** `financetoolkit`
- **Further reading:** Goodwin (1998)

6. M^2 Sortino Ratio (Part 1)

- **Formula:** $M_S^2 = (R_p - R_f) \cdot \frac{\sigma_M}{DD} + R_f$
- **Explanation:** Converts the Sortino Ratio into a return percentage format by adjusting for downside deviation instead of total volatility.
- **Purpose:** Allows more intuitive comparison of downside-risk-adjusted performance with benchmarks.
- **Usage:** Preferable for portfolios sensitive to negative returns.
- **Data Requirements:** Portfolio return, downside deviation, market volatility, and risk-free rate.

6. M^2 Sortino Ratio (Part 2)

- **Strengths:**
  - Same interpretability as M^2, but accounts for downside risk
  - Focused on risk that investors typically care most about
- **Limitations:**
  - Less widely reported than M^2 (Sharpe)
  - Requires correct estimation of downside deviation and market volatility
- **R function:** `M2Sortino()`
- **Python:** custom
- **Further reading:** Sortino and Price (1991)

7. Appraisal Ratio (Part 1)

- **Formula:** Appraisal Ratio $= \frac{\alpha}{\sigma_\epsilon}$
- **Explanation:** Measures a manager's alpha (excess return) relative to the volatility of residual (non-systematic) risk.
- **Purpose:** Evaluates manager skill independent of market movement.
- **Usage:** Applied in multi-factor and CAPM-adjusted assessments.
- **Data Requirements:** Regression alpha and residual standard deviation.

7. Appraisal Ratio (Part 2)

- **Strengths:**
  - Focuses purely on idiosyncratic risk-adjusted alpha
  - Useful complement to Information Ratio
- **Limitations:**
  - Requires accurate regression
  - Less common in everyday use
- **R function:** `AppraisalRatio()`
- **Python:** custom via `alpha / std(error)`
- **Further reading:** Roll (1992)

8. Beta (Part 1)

- **Formula:** $\beta = \frac{\text{Cov}(R_p, R_b)}{\text{Var}(R_b)}$
- **Explanation:** Measures the sensitivity of a portfolio's return to the benchmark (market) return.
- **Purpose:** Captures systematic (market) risk exposure.
- **Usage:** Risk decomposition, CAPM modeling, stress testing.
- **Data Requirements:** Portfolio and benchmark return series.

8. Beta (Part 2)

- **Strengths:**
  - Simple and widely understood
  - Used in numerous pricing and risk models
- **Limitations:**
  - Assumes linearity and stationarity
  - Beta may vary over time
- **R function:** `CAPM.beta()`
- **Python:** `financetoolkit`
- **Further reading:** Sharpe (1964)

9. Jensen's Alpha (Part 1)

- **Formula:** $\alpha = R_p - [R_f + \beta(R_b - R_f)]$
- **Explanation:** Measures abnormal return above what is predicted by CAPM.
- **Purpose:** Quantifies active management performance.
- **Usage:** CAPM-based evaluation and performance attribution.
- **Data Requirements:** Returns of portfolio, benchmark, and risk-free rate.

9. Jensen's Alpha (Part 2)

- **Strengths:**
  - CAPM-consistent
  - Widely used in academia and practice
- **Limitations:**
  - Requires accurate beta estimation
  - Ignores other risk factors
- **R function:** `CAPM.alpha()`
- **Python:** `financetoolkit`
- **Further reading:** Jensen (1968)

10. Active Return (Part 1)

- **Formula:** $\text{AR} = R_p - R_b$
- **Explanation:** Measures the difference between portfolio and benchmark returns.
- **Purpose:** Quantifies how much value was added or lost relative to the benchmark.
- **Usage:** Core to performance reporting.
- **Data Requirements:** Return series of portfolio and benchmark.

10. Active Return (Part 2)

- **Strengths:**
  - Easy to calculate and interpret
  - Always applicable if benchmark defined
- **Limitations:**
  - Does not adjust for risk
  - Sensitive to benchmark selection
- **R function:** `Return.excess(Rp, Rb)`
- **Python:** custom `Rp - Rb`
- **Further reading:** Goodwin (1998)

11. Active Risk (Part 1)

- **Formula:** $\text{ARisk} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_{p,i} - R_{b,i})^2}$
- **Explanation:** Measures the standard deviation of the difference between portfolio and benchmark returns (i.e., tracking error).
- **Purpose:** Captures volatility of excess returns due to active decisions.
- **Usage:** Performance and risk attribution.
- **Data Requirements:** Time series of portfolio and benchmark returns.

11. Active Risk (Part 2)

- **Strengths:**
  - Simple to compute and interpret
  - Complements active return in active management analysis
- **Limitations:**
  - Does not capture direction of deviations
  - Can be misleading without context
- **R function:** `TrackingError()`
- **Python:** `financetoolkit`
- **Further reading:** Goodwin (1998)

12. Alpha Information Ratio (Part 1)

- **Formula:** $AIR = \frac{\alpha}{\text{Tracking Error}}$
- **Explanation:** Measures value added by active management relative to the active risk.
- **Purpose:** Assesses skill-adjusted performance.
- **Usage:** Enhances appraisal of active manager effectiveness.
- **Data Requirements:** Alpha and tracking error.

12. Alpha Information Ratio (Part 2)

- **Strengths:**
  - Integrates alpha with risk of deviation
  - Clear benchmark-adjusted performance measure
- **Limitations:**
  - Requires stable alpha and error estimation
  - Assumes benchmark is well-defined
- **R function:** Custom from `CAPM.alpha()` and `TrackingError()`
- **Python:** Custom: `alpha / tracking_error`
- **Further reading:** Litterman (2001)

13. Omega Excess Return Ratio (Part 1)

- **Formula:** $\text{OER} = \frac{\Omega_p}{\Omega_b}$
- **Explanation:** Ratio of Omega of portfolio to Omega of benchmark (Omega = gain/loss probability ratio).
- **Purpose:** Compares probability-weighted gain/loss relative to benchmark.
- **Usage:** Risk-sensitive relative performance.
- **Data Requirements:** Portfolio and benchmark return distributions.

13. Omega Excess Return Ratio (Part 2)

- **Strengths:**
  - Works under non-normal returns
  - Tail-sensitive comparison
- **Limitations:**
  - Sensitive to target return and tail events
  - Computationally intensive
- **R function:** `OmegaExcessReturn(R)`
- **Python:** custom
- **Further reading:** Keating & Shadwick (2002)

14. Beta Co-Moments (Part 1)

- **Formula:** Based on higher-order co-moments such as skewness and kurtosis between portfolio and benchmark.
- **Explanation:** Captures asymmetry and fat-tailed dependencies between asset and market returns.
- **Purpose:** Evaluate non-linear relationships and co-risk structures.
- **Usage:** Tail-risk-sensitive portfolio diagnostics.
- **Data Requirements:** Portfolio and benchmark returns.

14. Beta Co-Moments (Part 2)

- **Strengths:**
  - Accounts for skewness and kurtosis in return co-behavior
  - Useful in modeling fat-tailed financial data
- **Limitations:**
  - Interpretation may be complex
  - Requires more data than linear beta
- **R function:** `BetaCoMoments()`
- **Python:** custom via higher-order moment estimation
- **Further reading:** PerformanceAnalytics Manual

15. Conditional Drawdown at Risk (CDD) (Part 1)

- **Formula:** $CDD_\alpha = \mathbb{E}[\text{Drawdown} | \text{Drawdown} > \text{VaR}_\alpha]$
- **Explanation:** Expected value of drawdowns exceeding a certain confidence level.
- **Purpose:** Provides downside risk insights beyond maximum drawdown.
- **Usage:** Advanced portfolio risk assessment, tail risk analysis.
- **Data Requirements:** Time series of returns.

15. Conditional Drawdown at Risk (CDD) (Part 2)

- **Strengths:**
  - More informative than Max Drawdown
  - Focuses on adverse tail events
- **Limitations:**
  - Sensitive to threshold level and sample length
  - Requires robust drawdown modeling
- **R function:** `CDD()`
- **Python:** Custom
- **Further reading:** PerformanceAnalytics Manual

16. Co-Moments (Part 1)

- **Formula:** Co-skewness, co-kurtosis computed as higher-order cross-moments between asset and benchmark.
- **Explanation:** Evaluates asymmetric and tail-dependent risk characteristics.
- **Purpose:** Enhances understanding of non-linear dependencies in returns.
- **Usage:** Used in risk parity, factor modeling, tail risk diagnostics.
- **Data Requirements:** Time series of asset and benchmark returns.

16. Co-Moments (Part 2)

- **Strengths:**
  - Captures deeper structure beyond variance and beta
  - Relevant in asymmetric return distributions
- **Limitations:**
  - Computationally intensive
  - Requires large datasets for stable estimates
- **R function:** `CoMoments()`
- **Python:** Custom
- **Further reading:** PerformanceAnalytics Manual

17. Fama Beta (Part 1)

- **Formula:** $\beta = \frac{\text{Cov}(R_p, R_f)}{\text{Var}(R_f)}$
- **Explanation:** Derived from the Fama-French model; measures sensitivity to specific factors like market, SMB, HML.
- **Purpose:** Decomposes portfolio risk into multifactor exposures.
- **Usage:** Multifactor performance attribution.
- **Data Requirements:** Portfolio returns and factor returns (e.g., Fama-French).

17. Fama Beta (Part 2)

- **Strengths:**
  - Decomposes beta across style and market factors
  - Core in academic asset pricing models
- **Limitations:**
  - Requires reliable factor data
  - Interpretation is model-dependent
- **R function:** `FamaBeta()`
- **Python:** `financetoolkit`
- **Further reading:** Fama & French (1993)

18. Market Timing (Part 1)

- **Formula:** Jensen's alpha regression extended with a quadratic term for market timing
- **Explanation:** Detects convexity in return–market relationship; identifies timing ability
- **Purpose:** Separate selection skill from timing skill
- **Usage:** Advanced performance attribution
- **Data Requirements:** Portfolio and market returns

18. Market Timing (Part 2)

- **Strengths:**
  - Isolates timing effects from alpha
  - Recognized in regression-based evaluation
- **Limitations:**
  - Requires careful model specification
  - Interpretation may depend on squared term sign
- **R function:** `MarketTiming()`
- **Python:** custom
- **Further reading:** Treynor & Mazuy (1966)

19. M^2 Excess (Part 1)

- **Formula:** $M^2_{excess} = (R_p - R_b) \cdot \frac{\sigma_M}{\sigma_p}$
- **Explanation:** Same as M^2 but in excess return form over benchmark
- **Purpose:** Assesses outperformance normalized to market risk
- **Usage:** Enhances Sharpe interpretation in relative terms
- **Data Requirements:** Returns and standard deviations of portfolio and benchmark

19. M^2 Excess (Part 2)

- **Strengths:**
  - Easier peer comparison
  - Risk-scaled relative return
- **Limitations:**
  - Not intuitive without context
  - Requires accurate risk estimates
- **R function:** `MSquaredExcess()`
- **Python:** Custom
- **Further reading:** PerformanceAnalytics Manual

20. Net Selectivity (Part 1)

- **Formula:** Net Selectivity = Total Return - Market Return - (Residual + Systematic Risk Premiums)
- **Explanation:** Measures return attributed solely to selection skill, excluding timing or risk exposures.
- **Purpose:** Isolate true selection effect in active management.
- **Usage:** Manager evaluation and performance attribution.
- **Data Requirements:** Returns and risk decomposition.

21. Net Selectivity (Part 2)

- **Strengths:**
  - Clear isolation of manager skill
  - Removes confounding timing or factor effects
- **Limitations:**
  - Depends on accuracy of risk decomposition
  - Can be affected by estimation errors
- **R function:** `NetSelectivity()`
- **Python:** custom
- **Further reading:** PerformanceAnalytics Manual

21. Selectivity (Part 1)

- **Formula:** Selectivity = Return - Expected Return based on factor exposures
- **Explanation:** Measures the degree to which performance is due to security selection rather than risk exposure.
- **Purpose:** Decomposes active return.
- **Usage:** Performance attribution framework.
- **Data Requirements:** Return series and factor model estimates.

22. Selectivity (Part 2)

- **Strengths:**
  - Isolates return due to selection
  - Integrates into multifactor frameworks
- **Limitations:**
  - Sensitive to factor specification
  - May overlap with alpha interpretation
- **R function:** `Selectivity()`
- **Python:** custom
- **Further reading:** PerformanceAnalytics Manual

22. SFM.alpha (Part 1)

- **Formula:** Intercept from regression of portfolio on benchmark
- **Explanation:** Alpha estimate from Single-Factor Model (CAPM)
- **Purpose:** Captures excess return unexplained by benchmark
- **Usage:** Basic active return estimation
- **Data Requirements:** Portfolio and benchmark returns

23. SFM.alpha (Part 2)

- **Strengths:**
  - Simple and interpretable
  - Common reference for alpha estimation
- **Limitations:**
  - Ignores multi-factor effects
  - Assumes constant beta and linearity
- **R function:** `SFM.alpha()`
- **Python:** custom from `financetoolkit`
- **Further reading:** PerformanceAnalytics Manual

24. SFM.beta (Part 1)

- **Formula:** Slope coefficient from regression of portfolio return on benchmark return
- **Explanation:** Beta estimate from the Single-Factor Model (CAPM)
- **Purpose:** Measures systematic risk relative to market
- **Usage:** Risk attribution and CAPM modeling
- **Data Requirements:** Time series of portfolio and benchmark returns

25. SFM.beta (Part 2)

- **Strengths:**
  - Core risk metric in finance
  - Easy to compute and interpret
- **Limitations:**
  - Assumes stable linear relationship
  - Ignores non-market risk factors
- **R function:** `SFM.beta()` from PerformanceAnalytics
- **Python:** Regression slope of `R_p` `R_b`
- **Further reading:** PerformanceAnalytics Manual

26. Specific Risk (Part 1)

- **Formula:** Variance of residuals from factor model regression
- **Explanation:** Measures the risk specific to the portfolio not explained by systematic factors
- **Purpose:** Captures idiosyncratic or non-factor risk
- **Usage:** Factor-based portfolio construction and evaluation
- **Data Requirements:** Factor model regression residuals

27. Specific Risk (Part 2)

- **Strengths:**
  - Identifies unhedged, asset-specific risk
  - Useful for diversification analysis
- **Limitations:**
  - Depends on accuracy of factor model
  - May not be stable over time
- **R function:** `SpecificRisk()`
- **Python:** custom via variance of regression residuals
- **Further reading:** PerformanceAnalytics Manual

28. Systematic Risk (Part 1)

- **Formula:** Variance explained by regression model (typically $\beta^2 \cdot \text{Var}(R_m)$)
- **Explanation:** Measures risk related to market movements or common factors
- **Purpose:** Quantifies exposure to systematic influences
- **Usage:** Risk budgeting, portfolio construction
- **Data Requirements:** Beta and market return variance

29. Systematic Risk (Part 2)

- **Strengths:**
  - Distinguishes risk from market exposure
  - Fundamental to CAPM and multifactor models
- **Limitations:**
  - Assumes model specification is correct
  - Ignores portfolio-specific risks
- **R function:** `SystematicRisk()`
- **Python:** custom from `financetoolkit`
- **Further reading:** PerformanceAnalytics Manual

30. Total Risk (Part 1)

- **Formula:** Total Risk $= \text{Var}(R_p)$
- **Explanation:** Total variance of portfolio returns
- **Purpose:** Captures both systematic and specific risk
- **Usage:** Baseline volatility measure in risk attribution
- **Data Requirements:** Portfolio return series

31. Total Risk (Part 2)

- **Strengths:**
  - Simple, comprehensive risk measure
  - Benchmark for comparing other risk metrics
- **Limitations:**
  - Doesn't distinguish risk sources
  - Doesn't reflect downside or tail risk
- **R function:** `TotalRisk()`
- **Python:** `np.var(Rp)` or `np.std(Rp)**2`
- **Further reading:** PerformanceAnalytics Manual

32. Up/Down Ratios (Part 1)

- **Formula:** `UpCapture`, `DownCapture`, `UpDownRatio` = `UpCapture / DownCapture`
- **Explanation:** Measures the portfolio's relative performance during up and down market periods. Calculate metrics on how the asset in R performed in up and down markets, measured by periods when the benchmark asset was up or down.
- **Purpose:** Captures asymmetry in return behavior during different market conditions.
- **Usage:** Bull/bear phase sensitivity, defensive/aggressive strategy classification.
- **Data Requirements:** Portfolio and benchmark returns.

32. Up/Down Ratios (Part 2)

- **Strengths:**
  - Interpretable and visually intuitive
  - Highlights defensive or offensive bias
- **Limitations:**
  - Ignores volatility and magnitude of deviation
  - Depends on clear market condition identification
- **R function:** `UpDownRatios()`
- **Python:** custom
- **Further reading:** PerformanceAnalytics Manual

Stochastic Dominance Efficiency (Part 1)

- **Concept:** An investment A stochastically dominates B if it yields higher utility for all (or a class of) investors.
- **Types:**
  - **FSD:** First-order stochastic dominance (for all risk-averse investors)
  - **SSD:** Second-order stochastic dominance (accounts for mean-variance preferences)
  - **TSD:** Third-order stochastic dominance (**accounts for mean-variance and positively skewed preferences**)
- **Purpose:** Identify investments superior across utility functions, without specifying a risk measure.
- **Usage:** Comparing cumulative return distributions, ranking portfolios.
- **Data Requirements:** Full empirical distribution of returns.

Stochastic Dominance Efficiency (Part 2)

- **Strengths:**
  - Model-free, non-parametric method
  - Captures full distributional differences
- **Limitations:**
  - Interpretation requires empirical or parametrical CDFs
  - No scalar summary statistic unless dominance gaps are measured
- **R packages:** `tsd()`, `stochdom`, `StochDominance`
- **Python:** Custom empirical CDF comparison or use `econtools`, `scipy.stats`
- **Further reading:** Levy (1992), StochDominance package

Stochastic Dominance Ratio (Part 1)

- **Definition:** The SD Ratio quantifies how far a portfolio is from violating stochastic dominance (SD) of a given order.
- **Conceptual Basis:** Compares how many times asset dominates and is dominated by other assets under different SD and distributional assumptions.
- **Formula:**

$$SD\ Ratio_i = \frac{D_i^+ - D_i^-}{D_i^+ + D_i^-} \quad \text{or} \quad R_i = \frac{D_i^+ - D_i^-}{\max\limits_{j \in \{1,\ldots,N\}} \{D_j^+ + D_j^-\}}$$

- **Purpose:** Converts binary pairwise dominance tests into interpretable efficiency scores (-1 to 1).

Stochastic Dominance Ratio (Part 2)

- **Interpretation:**
  - SD Ratio $= 1$: asset is efficient or is not dominated;
  - SD Ratio $= -1$: asset is dominated in all pairs where dominance was detected
  - Intermediate values: Degree of domination
- **Strengths:**
  - Works across SD orders (FSD, SSD, TSD)
  - May incorporate various distributions
  - Provides actionable ranking between portfolios
- **Limitations:**
  - For SSD requires to integrate all CDFs of all assets once, for TSD requires to integrate CDFs twice
  - Sensitive to sample size, time window, list of assets used and tail behavior / distributional assumptions

References:

- A. Kabašinskas, K. Šutienė, M. Kopa, K. Lukšys, K. Bagdonas. Dominance-Based Decision Rules for Pension Fund Selection under Different Distributional Assumptions, Mathematics, 2020
- M. Kopa, A. Kabašinskas, K. Šutienė. A stochastic dominance approach to pension-fund selection, IMA Journal of Management Mathematics, 2022
- A. Kabašinskas, K. Šutienė, M. Kopa. Robust evaluation of Baltic pension funds using Dynamic Stochastic Dominance Ratio, *under 2nd revision* in Journal of the Operational Research Society, 2025