# Week 5: Robo-Advisors & Portfolio Optimisation

# Learning Objectives

- Contrast traditional advisory and robo-advisory models, including the economics of access and cost
- Implement and interpret portfolio optimisation using Modern Portfolio Theory
- Explain why MPT breaks down at scale and in the presence of real-world constraints
- Describe how evolutionary algorithms extend portfolio optimisation to large-scale, multi-objective problems
- Evaluate recent frontier research connecting algorithmic innovation to robo-advisory applications

# Agenda

**Part I** : Economics of traditional vs. robo-advisory services
**Part II** : Technology and algorithms: Modern Portfolio Theory in practice
**Part III** : When theory meets reality: the hard problems in portfolio construction **Part IV** : The algorithmic revolution: from classical methods to frontier research **Part V** : Python implementation and directed learning

# Part I : Economics of Traditional vs. Robo-Advisory

# Traditional Wealth Management: The Service

**What financial advisers provide:**

- Portfolio management: asset allocation, security selection, rebalancing
- Financial planning: retirement, tax, estate planning advice
- Relationship management: ongoing consultations, market updates
- Bespoke service tailored to individual circumstances

High-touch, human-centred model built on trust and expertise

# Traditional Model: Cost Components

**Why is advice expensive?**

1. **Human Capital**: High salaries for credentialed staff (CFA, CFP)
2. **Technology**: Expensive per-seat licenses (Bloomberg, Aladdin)
3. **Compliance**: Regulatory filings, audits, liability insurance
4. **Overhead**: Physical offices, support staff

**Hilpisch Insight (Hilpisch (2019)):**
"Investments of $25-36 million just for derivatives analytics libraries" : High fixed costs favour large scale.

# Traditional Model: The Capacity Constraint

**The Binding Constraint: Time**

- **Max Capacity**: 100–150 relationships per adviser
- **Service requirements**: Quarterly meetings, bespoke planning, emotional support
- **Scaling**: To double clients, you must double advisers (Linear scaling)

**Unit Economics:**
Adviser Cost (£100k) / 100 Clients = **£1,000 per client cost**
*Before overheads or profit.*

# The Access Barrier: Economics of Exclusion

**Why Minimums Exist**

- **Break-even Point**: ~£100k assets (at 1.5% fee)
- **Profit Margin**: Firms need 2-3x break-even to be viable
- **Result**: Minimums set at **$250,000 – $500,000**

**Household Finance Reality (Campbell (2006)):**
The vast majority of households fall below this threshold.
*Median UK financial wealth: ~£15k.*

# The Access Barrier: Who is Left Out?

**The "Advice Gap"**

1. **Young Professionals**: High income, low assets (building wealth)
2. **Middle Class**: £20k–£150k savings (too small for advisers, too complex for DIY)
3. **Mass Market**: <£20k savings (relies on cash/deposits)

**Consequences:** - Poor diversification - High fees in mutual funds - Behavioural errors (panic selling) - **Exacerbated wealth inequality**

# Robo-Advisory: Automating the Value Chain

| Task | Traditional | Robo-Adviser |
|---|---|---|
| **Risk Assessment** | 1-hour interview | 5-min questionnaire |
| **Allocation** | Manual / spreadsheet | Mean-Variance Optimisation |
| **Rebalancing** | Quarterly (manual) | Daily (automated drift check) |
| **Tax Harvesting** | Rare (too complex) | Continuous (algorithmic) |

# Why Automation Raises the Stakes

**Tax-loss harvesting at scale**

Algorithms scan every holding daily, sell losers, and immediately buy a correlated substitute to capture the tax benefit without disrupting the portfolio. A human adviser managing 500 clients cannot do this; a robo-adviser manages 500,000 at the same marginal cost.

**Millisecond rebalancing**

Drift from target allocation is corrected continuously, not quarterly. The portfolio always reflects the optimised weights rather than drifting away through price movements.

⚠ The catch

Both advantages depend entirely on the quality of those target weights. If the underlying optimisation is unstable (producing extreme, erratic allocations), daily automated rebalancing **amplifies** the problem rather than solving it. This is the central challenge we examine for the rest of today.

# Robo-Advisory: Platform Scalability

**Software Economics (Zero Marginal Cost)**

- **Infrastructure**: Cloud-based (AWS), API-driven trade execution
- **Capacity**: 1 million clients served as easily as 1,000
- **Cost Curve**: High fixed cost (dev), near-zero marginal cost

**Comparison to Platforms (Week 5):**
Robo-advisers have **weak network effects** (my return doesn't depend on you) but **extreme scale economies**.

# Robo-Advisory: Fee Compression

**The New Price Point**

- **Traditional**: 1.0% – 1.5% AUM
- **Robo-Adviser**: 0.15% – 0.35% AUM
- **Savings**: ~**80% reduction**

**Access for All:** Minimums drop from **$250,000** to **$0 – $500**.

# Robo-Advisory: The Power of Compounding

**Impact of Fees on £100k Portfolio (30 Years, 6% Return)**

| Fee Model | Annual Fee | Final Wealth | Wealth Lost |
|---|---|---|---|
| **Traditional (1.5%)** | £1,500+ | £328,000 | **£246,000** |
| **Robo (0.25%)** | £250+ | £505,000 | £69,000 |

**Result:**
The traditional client loses **35% of potential wealth** to fees.

## Cost Comparison: UK Market (2024)

| Provider | Type | Annual fee | Minimum |
|---|---|---|---|
| Vanguard Personal Pension | Robo / passive | 0.15% | £500 |
| Nutmeg | Robo / managed | 0.45–0.75% | £500 |
| Wealthify | Robo / managed | 0.60% | £1 |
| St James's Place | Traditional IFA | ~1.5–2.0% | £20,000+ |
| Typical IFA (independent) | Traditional | 1.0–1.5% + initial 1–3% | £50,000–£250,000 |

*Sources: provider websites, FCA Retail Investments Data (2024), Reher and Sokolinski (2024).*

The structural point is straightforward: robo fees sit at 0.15–0.75%, traditional IFA fees

# Philippon's Puzzle: the Robo Exception

**Finance unit costs: stuck at ~2% for 130 years** (Philippon 2016)

*Why didn't technology reduce costs, as in every other industry?*

- Labour per client cannot be easily automated away
- Regulatory barriers limit new entrants
- Consumers can't observe quality $\rightarrow$ price competition is weak

**Robo-advisers break the mechanism**

Replacing human labour *per client* with software that scales at near-zero marginal cost is a **structural** change, not just cheaper service.

# Access Expansion or Two-Tier System?

**What the evidence shows** (Reher and Sokolinski 2024)

- 15–25% of new robo-adviser clients were *previously unadvised*: genuine access expansion, not substitution
- Largest gains for households with £10k–£100k in investable assets, precisely the segment excluded by traditional IFAs
- Diversification and asset allocation improve substantially for new entrants

**What automation cannot replace**

- Behavioural coaching during market panics (the adviser who stops you selling in March 2020)
- Holistic planning: debt, pension, property, insurance, tax together
- Adaptation to genuinely complex life situations

**!** The open question

Does this create *better access to a good product*? Or *algorithmic advice for those who cannot afford human advice*, whilst the wealthy retain personalised service? The evidence on welfare is the subject of Part III.

# Part II : Technology and Algorithms

# Modern Portfolio Theory

**Harry Markowitz (1952):** Investors care about return and risk; optimal portfolios balance these.

**Key insights:**

- **Diversification reduces risk**: Combining uncorrelated assets lowers portfolio volatility without sacrificing return
- **Efficient frontier**: Set of portfolios with highest return for each risk level
- **Optimal portfolio**: Depends on investor risk aversion (risk-return trade-off)

**Robo-adviser application:** Automate Markowitz optimisation using client risk tolerance, expected returns, and covariance matrix.

# The Risk-Return Trade-Off

The most fundamental idea in all of finance: **you cannot expect higher returns without accepting higher risk.** This single principle underpins the Capital Asset Pricing Model, the Efficient Market Hypothesis, and every portfolio construction method we study.

So why not simply pick the highest-returning asset?

A fund returning 15% with 40% volatility is **worse** than one returning 10% with 8% volatility. You are taking on far more uncertainty per pound of expected gain. Return alone is not the right objective.

# The Sharpe Ratio (Sharpe 1994)

$$\text{SR} = \frac{\bar{r} - r_f}{\sigma}$$

$\bar{r}$ = portfolio return, $r_f$ = risk-free rate (e.g. UK gilts), $\sigma$ = annualised volatility.

| SR | Interpretation |
|---|---|
| $< 0$ | Worse than holding cash |
| $0 - 0.5$ | Modest: typical of passive equity over long horizons |
| $0.5 - 1.0$ | Good: competitive with a diversified index fund |
| $> 1.0$ | Excellent: rare and likely fragile out-of-sample |

# Portfolio Optimisation: The Problem

Three inputs. One output.

| Symbol | Meaning | Source |
|---|---|---|
| $w$ | Weight vector: how much to allocate to each asset (what we choose) | Optimiser output |
| $\mu$ | Expected return vector: one forecast per asset | Historical data / model |
| $\Sigma$ | Covariance matrix: how assets move together | Historical data |
| $r_f$ | Risk-free rate (e.g. UK gilt yield) | Market |

$$\max_{w} \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma\, w}}$$

# Portfolio Return: The Easy Part

Portfolio return is a weighted average, nothing more:

$$r_p = w_1\mu_1 + w_2\mu_2 + w_3\mu_3 = w^\top \mu$$

**Concrete example:** 40% in Equities (8%), 30% in Bonds (4%), 30% in Real Estate (6%):

$$r_p = 0.40 \times 0.08 + 0.30 \times 0.04 + 0.30 \times 0.06 = 6.2\%$$

```
def portfolio_return(weights, expected_returns):
    return np.dot(weights, expected_returns)    # w'mu
```

# Portfolio Volatility: Where Diversification Lives

Portfolio volatility is **not** a weighted average of individual volatilities. For three assets it expands to:

$$\sigma_p = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + w_3^2\sigma_3^2 + 2w_1w_2\text{Cov}_{12} + 2w_1w_3\text{Cov}_{13} + 2w_2w_3\text{Cov}_{23}}$$

The cross-terms ($2w_iw_j\text{Cov}_{ij}$) are what matter. When stocks and bonds have low or negative covariance, holding both shrinks the portfolio's volatility below the average of the parts. That is diversification.

```
def portfolio_volatility(weights, cov_matrix):
    return np.sqrt(weights.T @ cov_matrix @ weights)   # sqrt(w'Sigma w)
```

The matrix multiplication $w^\top \Sigma w$ computes all $N^2$ variance and covariance terms in one operation, however many assets you have.

# Objective Function: A Coding Trick

`scipy.optimize.minimize` only *minimises*. To maximise the Sharpe ratio, we flip the sign:

```
def negative_sharpe(weights, expected_returns, cov_matrix, risk_free_rate)
    ret = portfolio_return(weights, expected_returns)
    vol = portfolio_volatility(weights, cov_matrix)
    return -(ret - risk_free_rate) / vol   # minimise this = maximise Shar
```

The solver finds the minimum of $-\text{SR}$, which is identical to finding the maximum of SR.

# Constraints: Keeping the Solution Sensible

Left unconstrained, the solver will make extreme bets, sometimes putting 80% in one asset. Two rules prevent that:

```
constraints = {'type': 'eq', 'fun': lambda w: np.sum(w) - 1}  # fully inves
bounds      = [(0, 0.40) for _ in range(n_assets)]            # no shorts,
```

| Constraint | What it enforces | Why it matters |
|---|---|---|
| Weights sum to 1 | Every pound is allocated | No cash drag or leverage |
| $w_i \geq 0$ | No short selling | Retail investors cannot short |
| $w_i \leq 0.40$ | No single position above 40% | Regulators and common sense |

# 3-Asset Example: Inputs

**Expected returns and volatilities**

| Asset | Return | Volatility |
|---|---|---|
| Stocks | 8% | 20% |
| Bonds | 4% | 10% |
| Real Estate | 6% | 17.3% |

```
exp_returns = np.array([0.08, 0.04, 0.06])
```

**Covariance matrix**

```
cov_matrix = np.array([
    [0.04,  0.010, 0.020],
    [0.010, 0.01,  0.005],
    [0.020, 0.005, 0.030]
])
```

Diagonal = variance ($\sigma^2$).

Off-diagonal = covariance. This is the key to diversification.

Convert to correlation: $\rho_{12} = \frac{\text{Cov}_{12}}{\sigma_1 \sigma_2}$

Stocks-Bonds: $\frac{0.010}{0.20 \times 0.10} = 0.50$

Bonds-RE: $\frac{0.005}{0.10 \times 0.173} = 0.29$ (better for diversification)

# Optimisation Results

```
import numpy as np
from scipy.optimize import minimize

def portfolio_return(weights, expected_returns):
    return np.dot(weights, expected_returns)

def portfolio_volatility(weights, cov_matrix):
    return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

def negative_sharpe(weights):
    ret = portfolio_return(weights, exp_returns)
    vol = portfolio_volatility(weights, cov_matrix)
    return -(ret - risk_free_rate) / vol if vol > 0 else np.inf

exp_returns    = np.array([0.08, 0.04, 0.06])
cov_matrix     = np.array([[0.04, 0.010, 0.020],
                           [0.010, 0.01, 0.005],
```

# What Is the Efficient Frontier?

We just found the best portfolio for a Sharpe-seeking investor. But what about a client who is 52, not 22?

The **efficient frontier** answers: for every level of volatility a client will accept, what is the highest return achievable? It is the full menu of optimal portfolios, not just one.

- A portfolio *on* the frontier cannot be improved without accepting more risk
- A portfolio *below* the frontier is dominated: diversification can do better at the same volatility

The robo-adviser questionnaire doesn't find a different portfolio for each client. It finds a different *point on the same curve*.

# The Capital Allocation Line

Once we identify the frontier, one portfolio stands out: the **tangency portfolio**, the point where a line drawn from the risk-free rate just touches the curve.

That line is the **Capital Allocation Line (CAL)**. Its slope is the Sharpe ratio of the tangency portfolio: the steepest achievable.

Every investor should hold the tangency portfolio as their *risky* allocation, then adjust overall risk by mixing it with the risk-free asset:

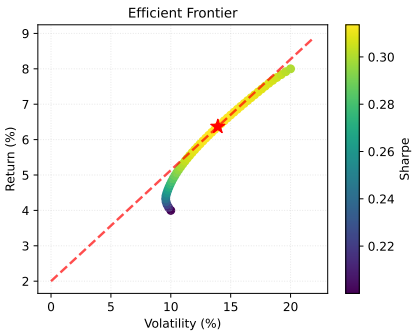
- **Conservative client**: 30% tangency portfolio + 70% gilts
- **Moderate client**: 70% tangency portfolio + 30% gilts
- **Risk-tolerant client**: 100% tangency portfolio (most robo-advisers stop here)

This is why robo-advisers produce the *same* underlying portfolio for all clients, then scale exposure up or down, not a different portfolio for each risk profile.

# Efficient Frontier Visualisation

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate efficient frontier
def generate_frontier(expected_returns, cov_matrix, risk_free_rate=0.02, n
    frontier_returns, frontier_vols, frontier_sharpes = [], [], []

    # Range of target returns
    min_ret = expected_returns.min()
    max_ret = expected_returns.max()
    target_returns = np.linspace(min_ret, max_ret, n_points)

    for target in target_returns:
        # Minimise volatility subject to target return
        n_assets = len(expected_returns)

        def portfolio_volatility(weights):
```

# Reading the Efficient Frontier

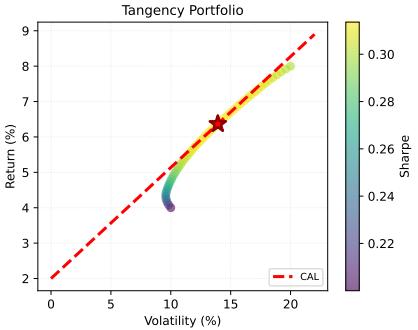



**What the curve tells us:**

- Each point = a different portfolio allocation
- Curve slopes upward: higher return requires higher risk
- Left side is steep: diversification is powerful here
- Right side flattens: diminishing returns to risk-taking
- Color: lighter yellow = higher Sharpe ratio

**Red star = optimal portfolio** (max Sharpe)

# The Optimal Portfolio




**Finding the best risk-adjusted return:**

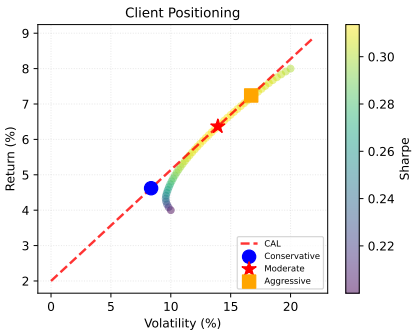
- **Tangency portfolio**: where Capital Allocation Line (red dashed) touches the frontier
- Maximises Sharpe ratio: (Return - Rf) / Volatility
- **Red star** marks the optimal point
- All investors should hold this portfolio (then adjust risk via mix with risk-free asset)

**Separation theorem:** Everyone holds the same risky portfolio, differing only in risk exposure

**The catch**

This is only true if expected returns and covariances are estimated correctly. We will see in Part III why that assumption is deeply problematic in practice.

# Client Choice and Risk Tolerance




**Mapping questionnaires to portfolios:**

- Robo-advisers assess risk tolerance via questionnaire
- Responses map to a position along the Capital Allocation Line
- **Conservative** (blue): more in risk-free assets (left)
- **Aggressive** (orange): more in tangency portfolio (right)
- Example: 25-year-old vs. 65-year-old retiree

**All hold same risky portfolio, differ in leverage**

# Implementation: Rebalancing and Tax-Loss Harvesting

**Maintaining and enhancing the portfolio:**

- **Rebalancing**: automatically sell winners, buy losers to maintain target allocation
- Prevents drift from chosen risk level
- Typical triggers: quarterly, or when asset drifts $>5\%$ from target
- **Tax-loss harvesting**: sell losing positions to generate tax deductions
- Buy similar (not identical) assets to maintain exposure
- Adds ~50-100 basis points to after-tax returns annually
- Can exceed robo-adviser fees (0.25-0.50%)

# Estimation Error: MPT's Achilles Heel

MPT assumes we *know* expected returns and covariances. We don't. We estimate them from noisy historical data, and the optimiser treats those estimates as facts.

- A 2pp revision to one asset's return estimate causes:

```
def optimize_portfolio(exp_ret, cov_mat, rf=0.02):
    n = len(exp_ret)
    def neg_sharpe(w):
        r = portfolio_return(w, exp_ret)
        v = portfolio_volatility(w, cov_mat)
        return -(r - rf) / v if v > 0 else np.inf
    result = minimize(neg_sharpe, np.ones(n)/n, method='SLSQP',
                      bounds=[(0, 1)]*n,
                      constraints={'type': 'eq', 'fun': lambda w: np.sum(w
    return result.x

base_returns = np.array([0.08, 0.10, 0.12])
sens_cov = np.array([[0.04, 0.01, 0.02],
```

# Solution 1: Bootstrap Weight Uncertainty

Point estimates for optimal weights give no sense of how reliable those weights are. Bootstrapping the return history shows the full range of "optimal" portfolios consistent with the data.

```
import pandas as pd

# Load real Bloomberg daily returns (SPY, BND, VNQ - 1,760 obs, 2018-2024)
_bdf = load_bloomberg()
_bdf['date'] = pd.to_datetime(_bdf['date'])
boot_daily = (_bdf[_bdf['ticker'].isin(['SPY','BND','VNQ'])]
              .drop_duplicates(subset=['date','ticker'])
              .pivot(index='date', columns='ticker', values='return')
              .dropna().sort_index())
boot_assets = boot_daily.columns.tolist()   # alphabetical: BND, SPY, VNQ
asset_labels = {'BND': 'BND (US Bonds)', 'SPY': 'SPY (US Stocks)', 'VNQ':

# Baseline optimal - unconstrained (0-100% per asset, no max position limi
ann_ret = boot_daily.mean().values * 252
```

# Solution 2: Out-of-Sample Validation

In-sample optimisation always looks good: the solver overfits to whatever history it sees. Walk-forward validation tests honestly by never letting the model see the data it is evaluated on.

The rolling-window procedure:

1. Estimate optimal weights on the first year of data
2. Hold that portfolio for the next quarter (out-of-sample)
3. Roll forward one quarter, re-estimate, repeat

```
import pandas as pd

# Load Bloomberg data: SPY, BND, VNQ (daily, 2018-2024)
_df = load_bloomberg()
_df['date'] = pd.to_datetime(_df['date'])
returns_history = (_df[_df['ticker'].isin(['SPY','BND','VNQ'])]
                   .drop_duplicates(subset=['date','ticker'])
                   .pivot(index='date', columns='ticker', values='return')
                   .dropna().sort_index())
```

# Solution 3: Bayesian Shrinkage

**Problem:** Sample means are noisy estimates of true expected returns

**Bayesian solution:** Shrink extreme estimates toward the grand mean (Week 1, §0.4)

**James-Stein estimator:**

$$\hat{\mu}_{\text{JS}} = \bar{\mu} + (1 - \lambda)(\hat{\mu}_i - \bar{\mu})$$

Notation: $\hat{\mu}_i$ = sample mean return for asset $i$, $\bar{\mu}$ = grand mean (average across assets).
$\lambda$ = shrinkage intensity (0 = no shrinkage, 1 = full shrinkage to grand mean).

```
# Sample estimates with one outlier (noisy estimation from a short history)
sample_returns = np.array([0.09, 0.18, 0.08])  # Bonds suspiciously high
js_cov = np.array([[0.04, 0.01, 0.02],
                   [0.01, 0.03, 0.005],
                   [0.02, 0.005, 0.03]])

grand_mean = sample_returns.mean()
shrinkage_lambda = 0.4
```

# Summary: When "Optimal" Breaks

What we saw with SPY/BND/VNQ and real data:

- **Sensitivity:** Small changes in one asset's return sent optimal weights to extremes (sensitivity slide).
- **Bootstrap:** "Optimal" SPY weight had a 90pp-wide 95% CI (10%–100%). Point estimates are not trustworthy.
- **Rolling backtest:** In our 7-year window the optimiser beat equal-weight; over many datasets and decades it usually loses (DeMiguel, Garlappi, and Uppal (2009)).
- **Shrinkage:** Pulling estimates toward the grand mean (Bonds down, Stocks/RE up) stabilised weights and avoided concentration.

Robo-advisers respond by constraining weights (e.g. 5–40% per asset), using shrinkage or equal-weight, and rebalancing to targets. They do not show clients that "optimal" weights have 90pp-wide uncertainty.

**The paradox**

MPT assumes we know expected returns and covariances. We only have noisy

# Real Correlation Structure

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = load_bloomberg()
df["date"] = pd.to_datetime(df["date"])
etfs = ["SPY", "TLT", "GLD", "QQQ", "EFA", "BND", "IWM", "VNQ"]
# Ticker → short description for axis labels
etf_labels = {"SPY": "SPY (US large cap)", "QQQ": "QQQ (US tech)", "IWM":
              "EFA": "EFA (ex-US developed)", "BND": "BND (US agg bonds)",
              "GLD": "GLD (gold)", "VNQ": "VNQ (real estate)"}
pivot = (df[df["ticker"].isin(etfs)]
         .drop_duplicates(subset=["date", "ticker"])
         .pivot_table(index="date", columns="ticker", values="log_return",
         .dropna())
corr = pivot.corr().rename(index=etf_labels, columns=etf_labels)
fig, ax = plt.subplots(figsize=(10, 8))
```

# What the Correlation Data Shows

- **Equity cluster:** SPY, QQQ, IWM, EFA correlate strongly with each other (about **0.76–0.93** in the heatmap); diversification among them is limited.
- **"Diversifiers" (TLT, GLD):** In this full-sample matrix, TLT is negative with equities (e.g. SPY), GLD is low positive; neither is a uniform safe haven. (Time variation comes on the next slide.)
- **8 assets** $\rightarrow$ **36** distinct entries in the covariance matrix (8 variances + 28 covariances); these estimates are noisy.
- Scale up: **500 assets** $\rightarrow$ **125,250** parameters ($500 \times 501/2$) from the same kind of history.

# Regime Change: When Diversification Fails

```
import pandas as pd
import matplotlib.pyplot as plt

_df2 = load_bloomberg()
_df2["date"] = pd.to_datetime(_df2["date"])
pivot = (_df2[_df2["ticker"].isin(["SPY", "TLT"])]
         .drop_duplicates(subset=["date", "ticker"])
         .pivot(index="date", columns="ticker", values="log_return")
         .dropna())
rolling_corr = pivot["SPY"].rolling(60).corr(pivot["TLT"]).dropna()
fig, ax = plt.subplots(figsize=(10, 3.5))
ax.plot(rolling_corr.index, rolling_corr.values, color="steelblue", linewi
ax.axhline(0, color="black", linewidth=0.8, linestyle="--")
ax.fill_between(rolling_corr.index, rolling_corr.values, 0,
                where=(rolling_corr.values > 0), alpha=0.3, color="red", la
ax.fill_between(rolling_corr.index, rolling_corr.values, 0,
                where=(rolling_corr.values <= 0), alpha=0.3, color="green"
```

# The 2022 lesson every optimiser missed:

- Pre-2022: bonds and equities *negatively* correlated: the standard diversification logic worked
- 2022: Federal Reserve raised rates aggressively; stocks *and* bonds fell simultaneously; correlation flipped positive
- Any MPT portfolio "optimised" using pre-2022 data suffered: the efficient frontier *itself* shifted
- The Markowitz assumption of a *stationary* covariance matrix is an assumption the market will eventually violate

# Part III : When Theory Meets Reality

# MPT's Hidden Assumptions

*The gap between elegance and reality*

**MPT requires us to know:**

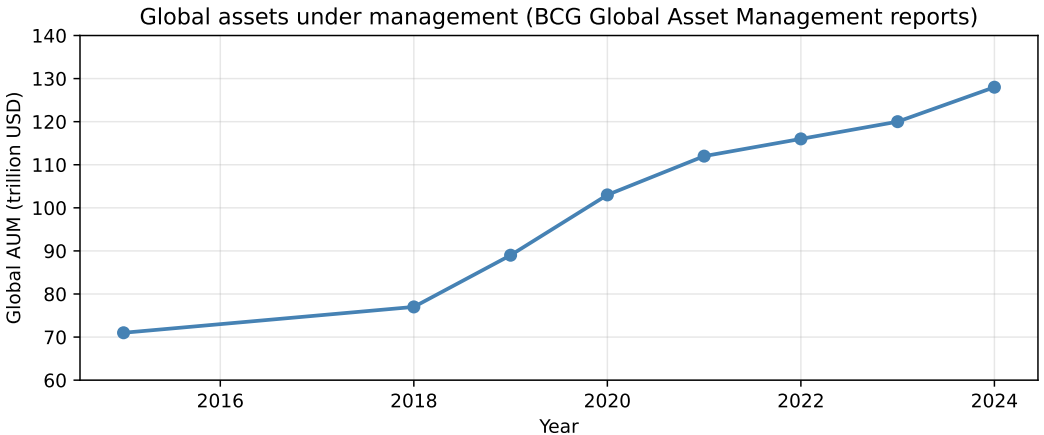
- Expected returns for every asset: estimated from noisy historical data
- The complete covariance matrix: $N(N+1)/2$ parameters for N assets
- That returns are approximately normally distributed
- That constraints are simple (weights sum to 1, no shorts)
- That the investment universe is small enough to solve exactly

**Reality:** None of these conditions reliably hold at scale

# The Scale Problem

*When size changes everything*

**Asset management at scale:**




- Global AUM reached **$128 trillion** in 2024 (12% growth); industry reports put it at

# The Scale Problem

**Why small-scale algorithms fail at large scale:**

- Classical quadratic programming solvers work well for ~100 assets
- At 1,000+ assets: computational cost explodes, estimation error dominates inputs
- Algorithms that reliably solve small problems often converge to poor solutions at large scale

The curse of dimensionality is not just a theoretical concern: it is the central practical challenge of modern asset management

# How Much of the Covariance Matrix Is Noise?

When we estimate a covariance matrix from historical data, not all of it is signal. Random Matrix Theory tells us exactly where the noise ends.

The Marcenko-Pastur Law gives the eigenvalue bounds expected from *purely random* returns:

$$\lambda_{\pm} = \sigma^2 \left(1 \pm \sqrt{\frac{M}{T}}\right)^2$$

where $M =$ assets, $T =$ observations, and $Q = T/M$ is the Q-ratio.

Any eigenvalue inside $[\lambda_-, \lambda_+]$ is statistically indistinguishable from noise.

# Why the Q-Ratio Changes Everything

$Q = T/M$ determines how much of the covariance matrix survives as genuine signal.

| Setting | $M$ assets | $T$ obs | $Q$ | Consequence |
|---|---|---|---|---|
| Bloomberg ETF lab | 8 | ~2,000 | 250 | Very few noise eigenvalues |
| Typical fund | 100 | 250 | 2.5 | Most eigenvalues are noise |
| Large-scale AM | 1,000 | 250 | 0.25 | Matrix is rank-deficient: uninvertible |

At $Q < 1$ there are fewer observations than assets and the covariance matrix cannot be inverted at all. **1–3 eigenvalues carry almost all genuine signal** in most estimated matrices.

# The Same Mathematics Powers AI

The noise-signal decomposition from portfolio theory reappears throughout modern machine learning.

- **Word embeddings** (GloVe, Word2Vec): a word co-occurrence matrix is factorised via SVD, the rectangular analogue of eigendecomposition. Keeping only the top-$k$ singular values is exactly the same logic applied to language.
- **Transformer fine-tuning** (LoRA): weight updates are written as $\Delta W = AB$ with rank $r \ll d$, explicitly discarding the noise subspace to learn efficiently from limited data.
- **Attention heads**: each head learns a different dominant eigenvector of the token-interaction matrix $QK^\top$, capturing a distinct semantic relationship.

We will return to this in the sequential learning week. For now, the core intuition is already in your hands.

# Denoising the Covariance Matrix: Three Steps

The RMT noise bound gives us a principled way to clean the covariance matrix rather than just shrinking it blindly.

1. **Decompose:** $\Sigma = V \Lambda V^\top$ (eigendecompose the sample covariance matrix)
2. **Classify:** eigenvalues *above* $\lambda_+$ carry genuine signal; those *below* are statistically noise
3. **Reconstruct:** replace noise eigenvalues with their mean, rebuild $\hat{\Sigma} = V \hat{\Lambda} V^\top$

The eigenvectors $V$ are preserved throughout: only the magnitude of the noise components is shrunk, not the direction of the correlations.

# The Eigenvalue Spectrum: Bloomberg 8-ETF Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

_df3 = load_bloomberg()
_df3["date"] = pd.to_datetime(_df3["date"])
etfs = ["SPY", "TLT", "GLD", "QQQ", "EFA", "BND", "IWM", "VNQ"]
pivot = (_df3[_df3["ticker"].isin(etfs)]
         .drop_duplicates(subset=["date", "ticker"])
         .pivot(index="date", columns="ticker", values="log_return")
         .dropna())
M, T = pivot.shape[1], pivot.shape[0]
cov = pivot.cov().values
eigenvalues, eigenvectors = np.linalg.eigh(cov)

sigma2 = np.mean(eigenvalues)
lam_plus = sigma2 * (1 + 1 / np.sqrt(T / M)) ** 2
```

# What the Denoised Matrix Achieves

Replacing noise eigenvalues with their mean produces a matrix that:

- Is **better conditioned** (no near-zero eigenvalues that blow up the inverse)
- Has the same **eigenvector structure** as the sample matrix (correlations are preserved)
- Is **theoretically justified**, unlike ad hoc shrinkage toward the grand mean

This is the same principle as James-Stein shrinkage, but with a mathematically grounded decision about *which* components to shrink rather than applying a uniform pull.

The result feeds directly into the optimiser. With Q  250, our 8-ETF universe already has a well-conditioned matrix; the denoising matters far more at scale (a typical fund has Q  2.5).

# Beyond Variance: Skewness Matters

MPT treats a +20% and a -20% return as identical risk. Investors do not.

- **Equity returns are negatively skewed:** crashes are more common than symmetry implies (2008, 2020).
- **Positive skewness commands a premium:** venture capital and options buyers pay for upside optionality.
- **Loss aversion amplifies the problem:** behavioural research shows losses hurt roughly twice as much as equivalent gains feel good.

Adding skewness as a third objective (maximise return, minimise variance, maximise skewness) converts the problem from a tractable quadratic programme into something far harder, with no single optimal solution.

# The Newly-Listed Securities Problem

MPT assumes every asset has an estimable return distribution. Large portfolios always include IPOs, recent listings, and newly opened markets where weeks or months of data offer little more than noise.

Excluding them misses high-growth opportunities. Including them requires **expert judgement**: formal beliefs about future performance drawn from fundamentals, sector knowledge, and analyst views.

The challenge is how to combine probability-based estimates and judgement-based estimates in a single coherent optimisation. This is precisely the problem Black-Litterman was designed to address.

# Real-World Constraints: From Quadratic to NP-Hard

Standard MPT needs only two constraints: weights sum to 1, no short-selling. Real portfolios add:

- **Cardinality:** hold between $K_{\min}$ and $K_{\max}$ securities; no infinitesimal positions
- **Minimum transaction lots:** invest in whole units (e.g. 100 shares per lot)
- **Boundary limits:** each security has a floor and a cap on allocation
- **Multiple objectives:** maximise return, minimise variance, and maximise skewness simultaneously

Adding cardinality and lot-size constraints converts the smooth quadratic programme into an NP-hard combinatorial problem. For $N = 500$ assets and $K = 50$ to hold, there are $\binom{500}{50} > 10^{62}$ possible portfolios: no exact algorithm can search them.

# The Gap: Textbook vs Reality

| | Textbook MPT | Real portfolio construction |
|---|---|---|
| Universe | Small, all well-established | Hundreds of assets, many newly listed |
| Constraints | Weights sum to 1, no shorts | Cardinality, lot sizes, boundary limits |
| Objectives | Minimise variance | Return, variance, skewness simultaneously |
| Solver | Exact quadratic programme | Approximation; NP-hard in general |
| Estimation | Assumed known | Noisy; some assets have no history |

Seventy years of research have been devoted to bridging this gap. Part IV traces that journey.

# Part IV : The Algorithmic Revolution

# Seventy Years of Portfolio Technology

| Generation | Era | Core approach | Where it breaks |
|---|---|---|---|
| **1st: Classical** | 1952–1990 | Markowitz mean-variance | Estimation error; breaks at scale |
| **2nd: Robust** | 1990–2010 | Shrinkage, Black-Litterman, factor models | Still single-objective, small universe |
| **3rd: Computational** | 2010– | Evolutionary algorithms, ML | Computationally intensive; active research |

Most retail robo-advisers today use 1st and 2nd generation methods. The research frontier, and the gap between theory and deployed products, sits in the 3rd.

# Black-Litterman: Market Consensus + Expert Views

Developed at Goldman Sachs in 1990, the intuition is Bayesian: start from what the market already believes, then layer on where you have genuine conviction.

- **Prior:** reverse-engineer expected returns from current market-cap weights via CAPM. This is the market's collective forecast, not noisy sample means.
- **Update:** "I believe UK equities will outperform by 2% over the next year." That view shifts the prior only in the assets it touches.
- **Result:** stable, well-diversified allocations; views only move weights where the manager has something to say.

The fix is elegant but still limited to small-to-medium portfolios with a single objective. It does not solve cardinality or lot-size constraints.

# When One Frontier Is Not Enough

The classical efficient frontier is two-dimensional: return versus variance. Add skewness as a third objective and the "frontier" becomes a surface.

On that surface there is no single "best" portfolio; there are only Pareto-optimal portfolios: ones where you cannot improve any objective without worsening at least one other. Think of a restaurant menu: the best value dish depends on what you care about (price, calories, taste).

The three objectives pull in different directions:

- Higher return generally requires more variance.
- Positive skewness (limited downside) often means accepting a lower expected return.
- An investor's personal weighting over the three determines which point on the surface is right for them.

The next generation of robo-advisers will match clients to a point on this Pareto surface, not just a single risk-tolerance band.

# Evolutionary Algorithms: Computing What Calculus Cannot

Classical optimisers need smooth, differentiable objectives. Cardinality and lot-size constraints destroy smoothness; gradient methods cannot navigate integer steps. The search space is too vast for exhaustive enumeration.

Evolutionary algorithms borrow from biology instead: maintain a *population* of candidate portfolios, score each against all objectives, then apply selection, crossover, and mutation. Better portfolios survive; the population evolves toward the Pareto surface.

**Multi-Objective Evolutionary Algorithms (MOEAs)** extend this idea to multiple simultaneous objectives. Rather than collapsing return, variance, and skewness into one number (like the Sharpe ratio), a MOEA maintains a diverse set of non-dominated solutions across the full Pareto frontier. No gradients are needed; integer constraints are handled naturally.

(Liu et al. 2024) demonstrate that MOEAs are the only class of algorithm that can handle NP-hard portfolio problems at scale.

# Why Scale Breaks Standard MOEAs

MOEAs work well at small scale (30 assets), but performance collapses at large scale (1,000 assets). Liu et al. (2024) (Fig. 1) show that *all* standard algorithms fail to approximate the Pareto frontier once the portfolio exceeds a few hundred securities.

The problem is **exploration versus convergence**: in thousands of dimensions, a population that searches broadly never converges; one that converges quickly gets stuck in a poor region. Standard algorithms cannot hold both.

Three research directions have emerged to fix this:

- **Group decision variables:** exploit the fact that assets in the same sector move together
- **Reduce the decision space:** use dimensionality reduction before optimising
- **Novel search strategies:** guide exploration toward promising regions of the Pareto surface

# Liu et al. (2025): Three Problems, One Framework

*IEEE Transactions on Evolutionary Computation*, Vol. 29, Feb 2025 (ABS 4) (Liu et al. 2024)

The paper takes the three open problems we have just covered and solves them simultaneously:

1. **Newly-listed securities:** an *uncertain random variable* framework blends probability-based estimates (established assets) with expert-judgement-based estimates (IPOs, recent listings) in one mathematically consistent model.
2. **Constraint handling:** an encoder-decoder mechanism transforms the NP-hard constrained problem (cardinality, lot sizes, boundaries) into an unconstrained one that any MOEA can address.
3. **Scale:** the LSWOEA algorithm combines decision space reduction with a dispersed target-guided search to maintain exploration-convergence balance at 1,000 assets.

# Results: Statistically Dominant at Every Scale

Tested against 9 benchmark MOEAs across 6 portfolio sizes (30 to 1,000 securities), LSWOEA achieves higher **hypervolume** on all six datasets. The performance advantage *grows* with scale, precisely where other algorithms fail most.

Hypervolume measures how much of the objective space is dominated by the Pareto frontier: unlike a single Sharpe ratio, it captures the quality of the entire trade-off surface. Statistical significance: Mann-Whitney U, Bonferroni-corrected, $p \ll 0.001$.

From any point on the resulting frontier, the investor can choose a strategy matching their preference:

- **Return preference:** higher expected return, accepting more variance and lower skewness
- **Risk preference:** minimum variance, larger allocation to the risk-free asset
- **Skewness preference:** positive asymmetry, accepting lower expected return

Running time: roughly **15 seconds** for a 1,000-security portfolio on standard hardware.

# From Today's Robo to Tomorrow's Algorithmic Engine

| Dimension | Today's robo-adviser | Tomorrow's system |
|---|---|---|
| Universe | 6–12 ETFs | Thousands of securities |
| Objectives | Sharpe ratio (2D) | Return, variance, skewness (3D Pareto) |
| New assets | Excluded or ad hoc | Uncertain random variable framework |
| Solver | Constrained MPT | MOEA at scale |
| Client interface | Risk band (1–10) | Point on Pareto surface |

The shift is from automating a 1952 model to operationalising 2025 research. The robo-adviser of tomorrow is an evolutionary engine running on the Pareto frontier.

# Part V : Python Implementation and Directed Learning

# Lab: Build Your Own Robo-Adviser

Five tasks, each mirroring a section of today's lecture:

1. Recreate the fee comparison and visualise access expansion
2. Build the portfolio optimiser; test on SPY, BND, VNQ
3. Generate the efficient frontier; identify the tangency portfolio
4. Perturb expected returns and see how weights respond
5. Reflect: what would you need to add to handle 1,000 assets and three objectives?

**Deliverable:** notebook with code, plots, and 300–400 word interpretation connecting code to theory.

# Quick Check-In Questions

**Q1:** Why do robo-advisers have lower minimum account sizes than traditional advisers?
**Q2:** What is the efficient frontier, and how do robo-advisers use it?
**Q3:** Name one reason why standard portfolio optimisation algorithms struggle with large-scale portfolios of 1,000+ securities.

# Assessment Touchpoints

**CW1: Business Analysis Presentation (Week 6)**

- **Task**: Analyse a FinTech business model (e.g., Nutmeg, Wealthfront).
- **Relevance**: Apply the economics of robo-advisory to explain their value proposition.
  - How do they lower costs?
  - Who is their target market?
  - What are the risks?

**CW2: Technical Implementation (Week 13)**

- **Task**: Build a portfolio optimisation tool.
- **Relevance**: The Python code we write today (Part V) is the core engine of a robo-adviser.
- **Key Skill**: Implementing MPT and backtesting performance.

# Directed Learning Plan ( 3 hours)

**Reading (60 min):**

- Hilpisch (2019) Chapter 13 (portfolio analytics and optimisation)
- Reher and Sokolinski (2024) (empirical evidence on access and welfare)
- Gu, Kelly, and Xiu (2020) Sections 1-2 (ML in asset pricing, optional but valuable)

**Practical (75 min):**

- Complete Lab 04 Tasks 1-5
- Include code, plots, and interpretations

**Reflection (45 min):**

- Write 300-400 words: "Who benefits most from robo-advisers? What are the main risks? How should governance evolve?"
- Use at least one citation

# Project preview: factor-based investing

**Preview of an end-of-semester project pathway:**

You'll replicate or extend research on factor-based investing using professional factor data

**Today's primer task (optional, 20-30 min):**

- Explore the Jensen-Kelly-Pedersen (JKP) factor dataset (`resources/jkp-sample.csv`)
- Understand what factors like MKT, SMB, HML, MOM represent
- Compute basic summary statistics and visualize cumulative returns
- Connect factors to robo-advisor portfolio construction

**Why connect this to robo-advisors?**

Modern robo-advisors don't just diversify across assets: they target systematic factor exposures (value, momentum, quality). Understanding factors is essential for evaluating algorithmic investment strategies.

# Looking Forward: From Characteristics to Embeddings

**Traditional approach:**
Robo-advisers use hand-crafted characteristics (size, value, momentum) to build portfolios

**Emerging approach:**
Learn asset relationships from portfolio holdings data: "asset embeddings"

**Key insight from recent research (Gabaix et al. 2025):**
Portfolio holdings encode rich information about which assets belong together

**Analogy:** Just as words appearing in similar contexts have related meanings, assets appearing in similar portfolios share investment characteristics

# Key Takeaways

**1.** Robo-advisers automate MPT, reducing marginal cost to near-zero and expanding access to the \$25K–\$150K wealth band (Reher and Sokolinski (2024)).

**2.** But estimation error is severe: a 90pp-wide 95% confidence interval on the "optimal" SPY weight (our Bloomberg data) is not a rounding error.

**3.** Solutions exist: rolling-window validation, Bayesian shrinkage, RMT denoising. Each addresses a specific failure mode.

**4.** At scale, NP-hard constraints and the exploration-convergence breakdown require a new generation of evolutionary algorithms (MOEAs).

**5.** The frontier (Liu et al. (2024)) is 15 seconds for 1,000 securities. The gap between academic research and deployed products is narrowing fast.

# References

See chapter bibliography for full citations.

Core readings:

- Hilpisch (2019) Chapter 13 : Portfolio analytics and optimisation
- Reher and Sokolinski (2024) : Empirical evidence on robo-adviser access and welfare
- Gu, Kelly, and Xiu (2020) : Machine learning in empirical asset pricing
- Vives (2019) : Digital disruption and FinTech taxonomy

# Slides Bibliography


Boston Consulting Group. 2025. "Global Asset Management 2025: The Industry Hit a New Record High in 2024." BCG report and press release. https://www.bcg.com/press/29april2025-global-asset-management-record-high-critical-turning-point.

Campbell, John Y. 2006. "Household Finance." *Journal of Finance* 61 (4): 1553–1604. https://doi.org/10.1111/j.1540-6261.2006.00883.x.

DeMiguel, Victor, Lorenzo Garlappi, and Raman Uppal. 2009. "Optimal Versus Naive Diversification: How Inefficient Is the 1/n Portfolio Strategy?" *The Review of Financial Studies* 22 (5): 1915–53. https://doi.org/10.1093/rfs/hhm075.

Gabaix, Xavier, Ralph S. J. Koijen, Robert Richmond, and Motohiro Yogo. 2025. "Asset Embeddings." Working Paper. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4507511.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2020. "Empirical Asset Pricing via Machine Learning." *Review of Financial Studies*. https://doi.org/10.1093/rfs/hhaa009.