Drawdown Risk: Measurement, Prediction, and Mitigation

Your Name

Table of contents

1 Executive Summary

Drawdowns — sustained declines from prior peaks — are a central measure of investment risk. They capture not only the size of losses but also their duration and the psychological and liquidity pressures they impose on market participants. While volatility remains the industry's standard risk metric, drawdowns often determine *actual* investor behavior, influencing capital withdrawals, mandate breaches, and risk appetite.

This study examines drawdown risk through three complementary lenses:

1. **Mechanics and measurement** — formal definitions of maximum drawdown (MDD), average drawdown, and time under water, along with their statistical properties under different return-generating processes.
2. **Prediction and early warnings** — the potential to forecast elevated drawdown risk using tools from both quantitative finance (e.g., correlation spikes, regime-switching probabilities, LPPL bubble detection) and market theory (e.g., Soros's reflexivity, Mandelbrot's multifractal markets).
3. **Mitigation strategies** — evaluating the performance of stop-loss rules, momentum-based overlays, and explicit hedges such as VIX futures and protective options.

A central theme is the **connection between stop-loss rules and momentum trading**. A trailing stop that re-enters only above the prior peak inherently captures part of a time-series momentum strategy: it cuts risk when recent returns are negative and adds exposure when trend resumes. This link suggests that the efficacy of stop-losses will vary with the degree of return autocorrelation and the persistence of regimes — a hypothesis we test both in simulations and on historical data.

Stop-losses are not the only path to drawdown control. We compare them with *explicit hedging* strategies, including: - **Volatility hedges**: buying VIX futures or calls when early-warning indicators trigger. - **Tail hedges**: purchasing out-of-the-money index puts or put spreads to cap downside. - **Dynamic allocation shifts**: moving capital to low-volatility or uncorrelated assets in high-risk regimes.

Our empirical analysis spans the S&P 500, US 10-year Treasury notes, and Bitcoin — assets with distinct volatility, correlation, and drawdown profiles. Simulation studies use AR(1), regime-switching, and stochastic volatility models to explore a wide parameter space and quantify expected performance differences across strategies.

Key takeaways (preview): - **Stop-loss effectiveness is conditional**: they tend to improve risk-adjusted returns when returns exhibit positive serial correlation and regimes persist, but can hurt performance in mean-reverting markets. - **Hedges offer complementary protection**: while they often cost carry in normal markets, they can sharply reduce maximum drawdown and time under water when timed with credible early-warning signals. - **Integrated approaches dominate**: blending signal-driven stop-loss logic with targeted hedging yields better drawdown control per unit of performance drag than either technique alone.

This report proceeds from measurement to prediction to mitigation, grounding each step in both simulation and real-world data. By the conclusion, we present a framework for active drawdown management that integrates statistical signals, behavioral insights, and cost-aware strategy design.

2 Definitions and Measurement

2.1 Why definitions matter

Consistent drawdown metrics allow fair comparison across strategies and time periods. While **Maximum Drawdown (MDD)** is common, it captures only the single worst episode. Complementary measures like **Average Drawdown (AvgDD)**, **Time Under Water (TUW)**, **Ulcer Index (UI)**, and **Recovery Time** help characterize the *shape* and *persistence* of losses.

2.2 Key metrics & formulas

2.2.1 Maximum Drawdown (MDD)

Definition: largest peak-to-trough loss.

$$\mathrm{MDD} = \min_{t} \left(\frac{P_t}{\max_{\tau \le t} P_\tau} - 1 \right)$$

2.2.2 Average Drawdown (AvgDD)

Definition: mean depth across all drawdowns.

$$\mathrm{AvgDD} = \frac{1}{N_{\mathrm{dd}}} \sum_{i=1}^{N_{\mathrm{dd}}} \left(\frac{P_{\mathrm{trough},i}}{P_{\mathrm{peak},i}} - 1 \right)$$

2.2.3 Time Under Water (TUW)

Definition: average duration from a peak to full recovery.

$$\mathrm{TUW} = \frac{1}{N_{\mathrm{dd}}} \sum_{i=1}^{N_{\mathrm{dd}}} \left(t_{\mathrm{rec},i} - t_{\mathrm{peak},i} \right)$$

2.2.4 Ulcer Index (UI)

Definition: RMS of percentage drawdowns (emphasizes persistent pain).

$$\mathrm{UI} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(100 \times \min\left(0, \frac{P_t}{\max_{\tau \le t} P_\tau} - 1\right)\right)^2}$$

2.2.5 Recovery Time

Definition: longest time from a peak to its recovery.

$$\max_i \left(t_{\mathrm{rec},i} - t_{\mathrm{peak},i}\right)$$

2.3 SPY example (downloaded with yfinance)

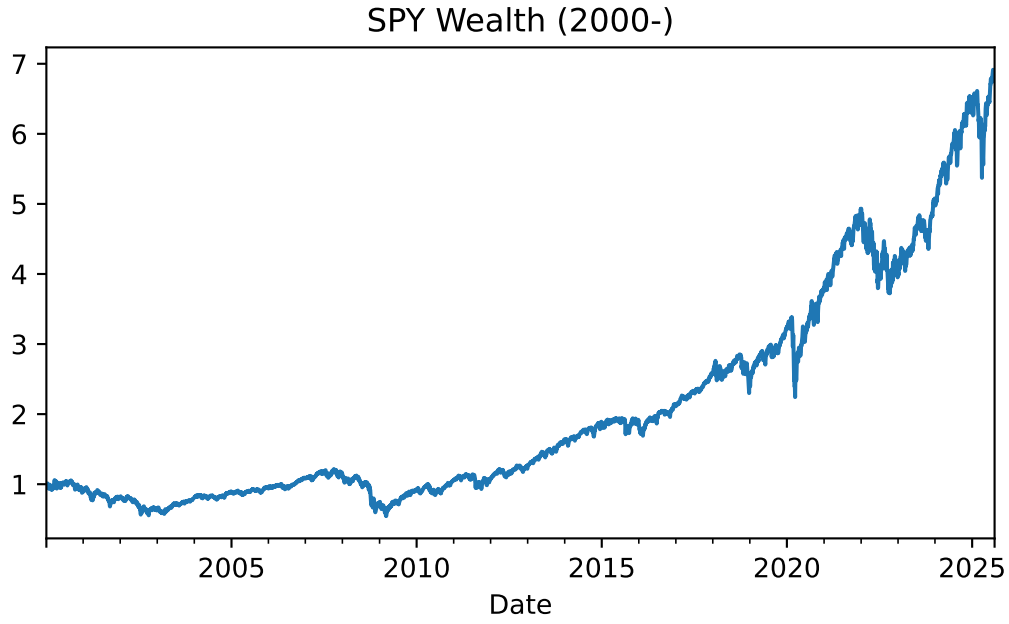
```
import pandas as pd, numpy as np, matplotlib.pyplot as plt
import yfinance as yf
from datetime import datetime

spy = yf.download("SPY", start="2000-01-01", auto_adjust=True, progress=False)["Close"].squee
rets = spy.pct_change().dropna()
wealth = (1+rets).cumprod()
dd = wealth/wealth.cummax() - 1

# Identify drawdown episodes for AvgDD and TUW
peaks = wealth.cummax()
in_dd = dd < 0
# Episode IDs increment when leaving water and re-entering
episode_id = ( (~in_dd).astype(int).diff() == -1 ).cumsum()
episode_id = episode_id.where(in_dd, np.nan).ffill()

draws = []
if episode_id.notna().any():
    for eid, grp in wealth.groupby(episode_id.dropna()):
        # grp spans underwater segment including starting day after a peak
        depth = (grp/grp.cummax() - 1).min()
        # Recovery time: from peak to first recovery beyond that peak
        start_idx = grp.index[0]
```

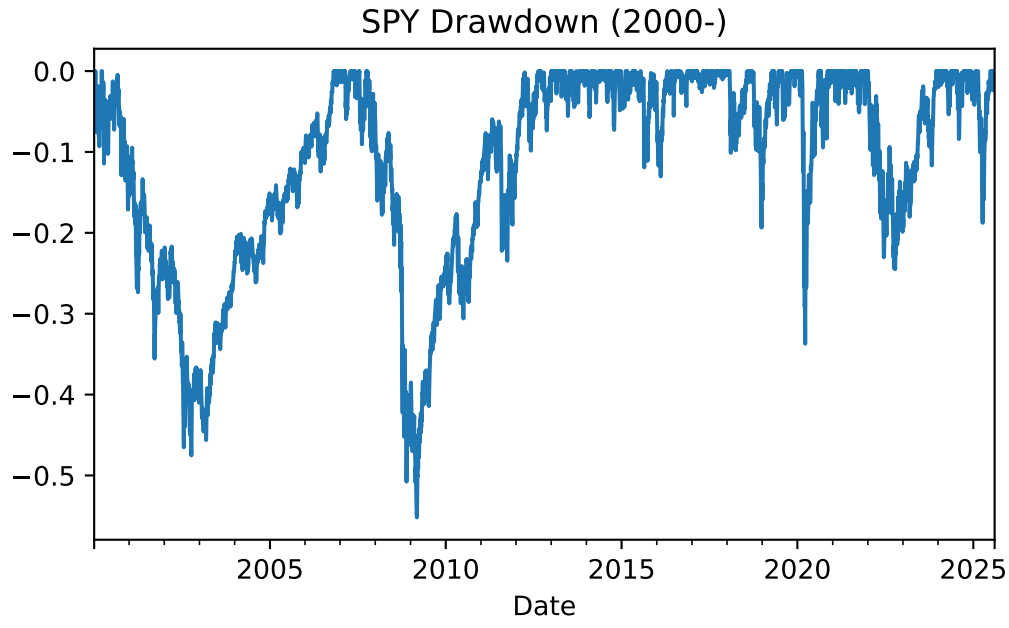
```
        peak_level = peaks.loc[start_idx]
        # find recovery
        after = wealth.loc[start_idx:]
        rec_idx = after[after >= peak_level].index.min()
        if pd.isna(rec_idx):
            rec_days = np.nan
        else:
            rec_days = (rec_idx - start_idx).days
        draws.append((depth, rec_days))

avgdd = float(np.nanmean([d for d,_ in draws])) if draws else np.nan
tuw = float(np.nanmean([u for _,u in draws])) if draws else np.nan
recovery_time = float(np.nanmax([u for _,u in draws])) if draws else np.nan

mdd = float(dd.min())
vol = float(rets.std()*np.sqrt(252))
cagr = float((wealth.iloc[-1]/wealth.iloc[0])**(252/len(wealth))-1)
ui = float(np.sqrt(( (100*np.minimum(0, dd)).pow(2) ).mean()))

tbl = pd.DataFrame({
    "CAGR":[cagr],
    "Vol":[vol],
    "Sharpe":[cagr/vol if vol else np.nan],
    "MDD":[mdd],
    "AvgDD":[avgdd],
    "TUW_days":[tuw],
    "RecoveryTime_days":[recovery_time],
    "UlcerIndex":[ui]
})
display(tbl.round(4))
import os
os.makedirs("figures", exist_ok=True)
tbl.to_csv("figures/spy_drawdown_metrics.csv", index=False)
print("Saved to figures/spy_drawdown_metrics.csv")
```

	CAGR	Vol	Sharpe	MDD	AvgDD	TUW_days	RecoveryTime_days	UlcerIndex
0	0.0772	0.1914	0.4035	-0.5519	-0.0169	31.0366	2404.0	16.4171

```
Saved to figures/spy_drawdown_metrics.csv
```

```
# Quick plots
plt.figure(); wealth.plot(title="SPY Wealth (2000-)"); plt.tight_layout(); plt.show()
plt.figure(); dd.plot(title="SPY Drawdown (2000-)"); plt.tight_layout(); plt.show()
```





2.4 Practical notes

- MDD focuses on a single tail event; AvgDD and UI provide stability for optimization.
- TUW and Recovery Time reflect investor experience and liquidity needs.
- In stop-loss testing, AvgDD and TUW often show larger improvements than MDD.

3 Literature Review

3.1 Drawdown theory and measures

- Brownian/diffusion results (Magdon-Ismail et al. (2004); Hadjiliadis and Vecer (2006)).
- CED and CDaR risk frameworks (Goldberg and Mahmoud (2014); Chekhlov, Uryasev, and Zabarankin (2005)).

3.2 Predicting drawdowns and crashes

- LPPL and critiques (Sornette (2003); Johansen, Ledoit, and Sornette (2000); Filimonov and Sornette (2013)).
- Correlation eigenvalue spikes (Conlon, Ruskin, and Crane (2010)).
- Regime-switching warnings (Lo and Remorov (2017)).
- Reflexivity (Soros) and multifractals (Mandelbrot).

3.3 Stop-Loss rules and momentum

- When do stops help? (Kaminski and Lo (2014))
- Serial correlation, regime persistence, and costs (Lo and Remorov (2017)).

4 Simulation Framework

We simulate AR(1) processes and evaluate Buy & Hold vs a 5% trailing stop, reporting CAGR, Vol, Sharpe, MDD with 95% confidence intervals (Monte Carlo).

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from scripts.simulations import simulate_ar1, trailing_stop_returns, metrics

np.random.seed(1)
years = 5
rho_grid = np.linspace(-0.5, 0.5, 9)
trials = 30
sigma_ann = 0.05
mu_ann = 0.0

def run_ci(grid, vary="rho"):
    rows = []
    for val in grid:
        bh, sl = [], []
        for _ in range(trials):
            price = simulate_ar1(mu_ann=mu_ann if vary=="rho" else val,
                                 rho=val if vary=="rho" else 0.0,
                                 sigma_ann=sigma_ann, years=years, seed=None)
            bh_rets = price.pct_change().dropna()
            sl_rets = trailing_stop_returns(price, stop_pct=0.05).iloc[1:]
            bh.append(metrics(bh_rets)); sl.append(metrics(sl_rets))
        for name, coll in [("BuyHold", bh), ("Stop5pct", sl)]:
            dfm = pd.DataFrame(coll)
            for m in ["CAGR","Vol","MDD","Sharpe"]:
                mu = dfm[m].mean(); se = dfm[m].std(ddof=1)/np.sqrt(len(dfm))
                rows.append({"Strategy":name, "x":val, "Metric":m,
                             "Mean":mu, "Low":mu-1.96*se, "High":mu+1.96*se})
    return pd.DataFrame(rows)

df = run_ci(rho_grid, vary="rho")

for m, ylab in [("CAGR","CAGR"),("Vol","Annualized Vol"),("MDD","Max Drawdown"),("Sharpe","S
```

```
    plt.figure()
    for strat in ["BuyHold","Stop5pct"]:
        sub = df[(df["Metric"]==m) & (df["Strategy"]==strat)].sort_values("x")
        plt.plot(sub["x"], sub["Mean"], label=strat)
        plt.fill_between(sub["x"], sub["Low"], sub["High"], alpha=0.2)
    plt.xlabel("rho"); plt.ylabel(ylab); plt.title(f"{ylab} vs rho ( =0)")
    plt.legend(); plt.tight_layout(); plt.show()
```

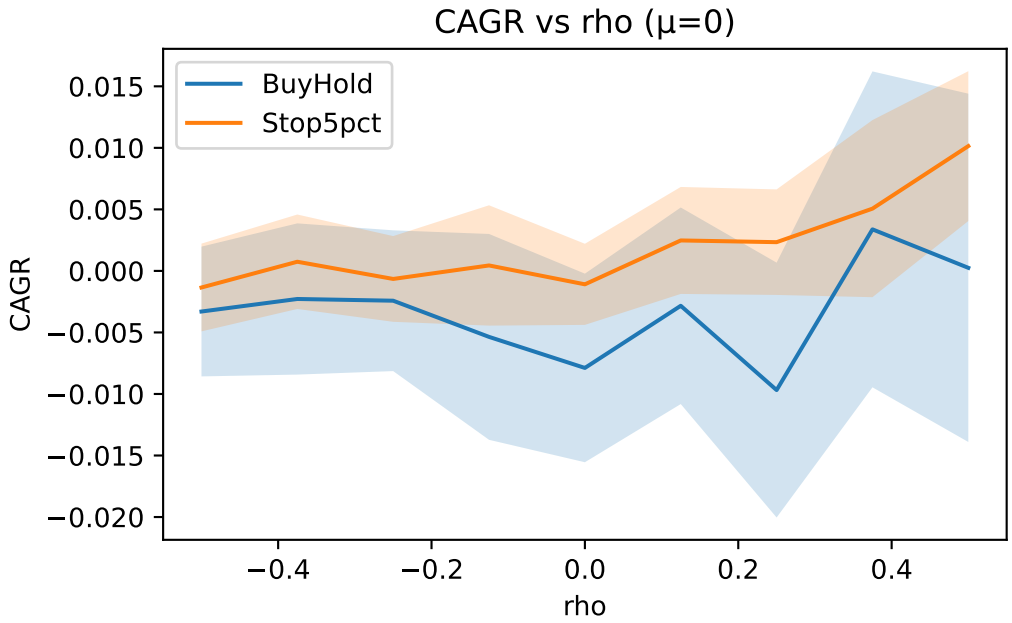
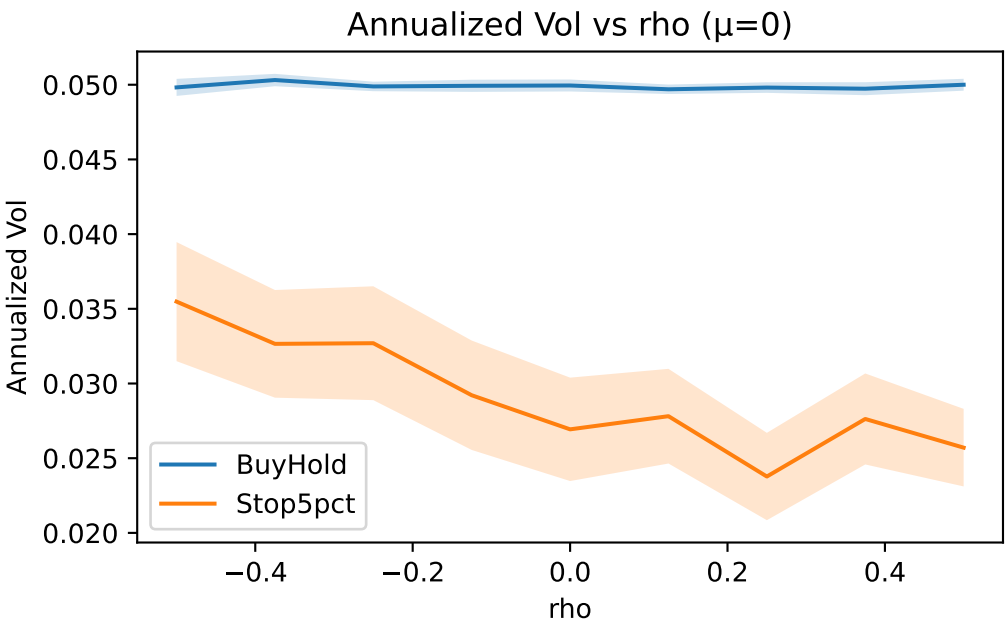
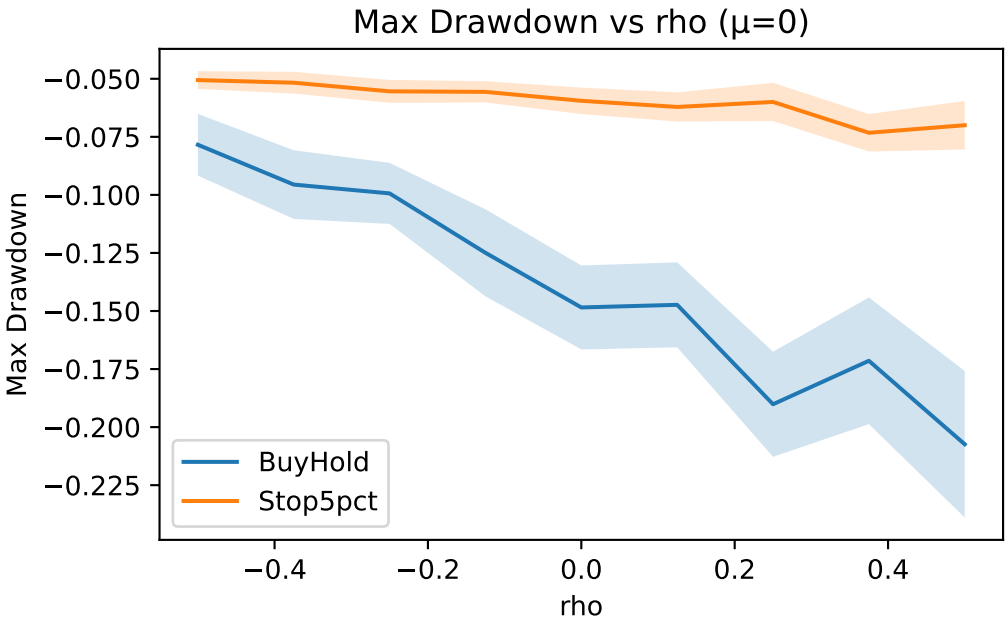
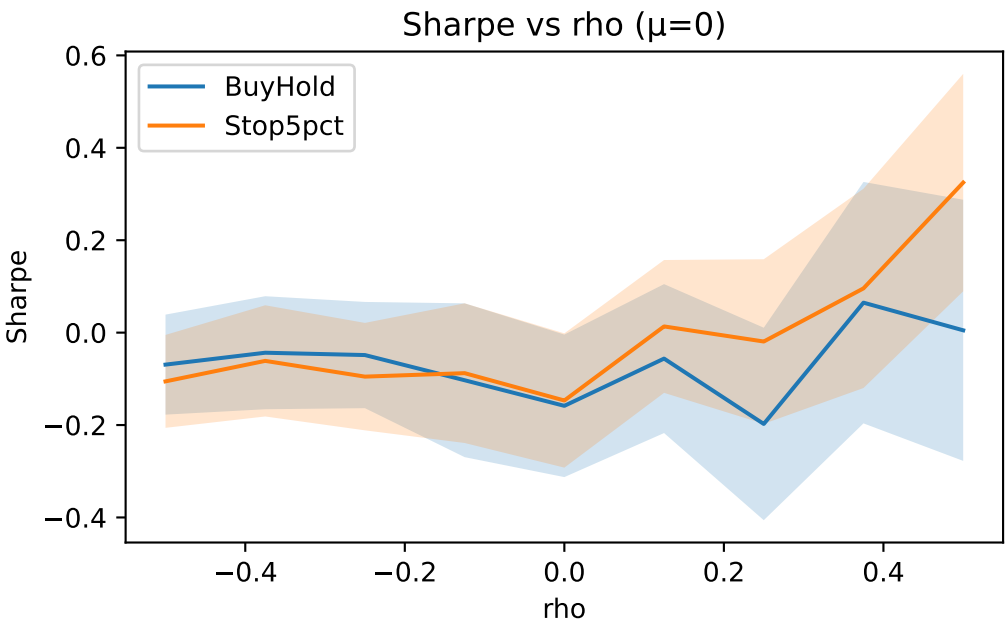


Figure 4.1: AR(1), =0: Metrics vs with 95% CI (example run)


Annualized Vol vs rho (μ=0)
Annualized Vol
0.050
0.045
0.040
0.035
0.030
0.025
0.020
−0.4
−0.2
0.0
0.2
0.4
rho
BuyHold
Stop5pct


Max Drawdown vs rho (μ=0)
Max Drawdown
−0.050
−0.075
−0.100
−0.125
−0.150
−0.175
−0.200
−0.225
−0.4
−0.2
0.0
0.2
0.4
rho
BuyHold
Stop5pct


Sharpe vs rho (μ=0)
BuyHold
Stop5pct
Sharpe
rho
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4

5 Empirical Measurement

We fetch daily data with **yfinance** and compute: returns, wealth & drawdowns, rolling / / , and a summary table for SPY, IEF, and BTC.

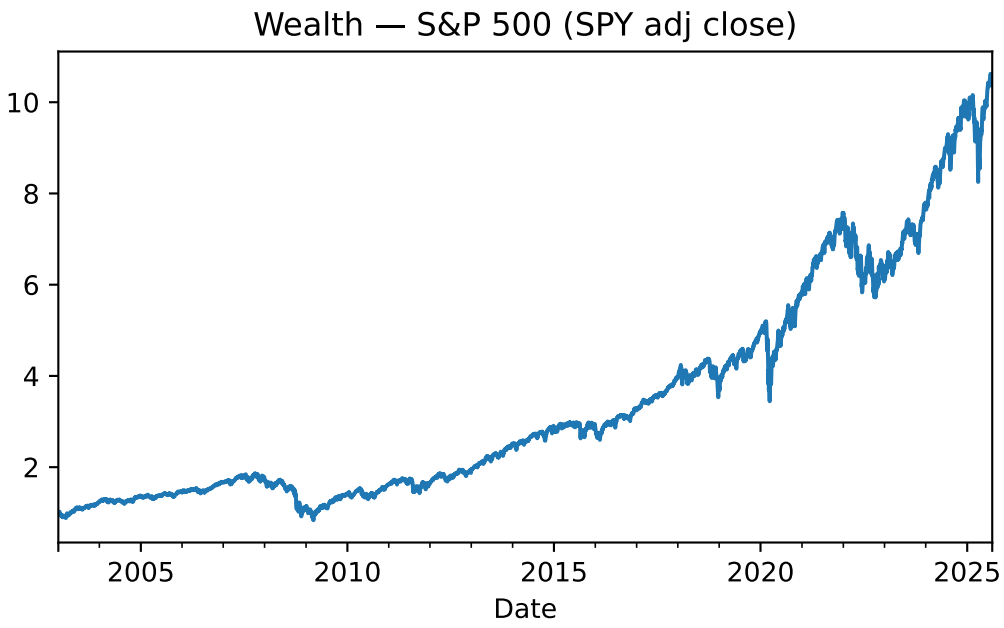
```
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from scripts.empirical_data import load_prices_yf, pct_returns, wealth_series, drawdown_serie

tickers = {
    "S&P 500 (SPY adj close)": "SPY",
    "US 7-10Y Treasury (IEF)": "IEF",
    "Bitcoin (BTC-USD)": "BTC-USD",
    "VIX Index (proxy only)": "^VIX"
}

prices = {}
for label, t in tickers.items():
    try:
        prices[label] = load_prices_yf(t, start="2003-01-01")
    except Exception as e:
        print(f"Skipping {label}: {e}")

rets = {k: pct_returns(v) for k, v in prices.items() if v is not None}
summ = {k: summary_metrics(r) for k, r in rets.items() if k != "VIX Index (proxy only)"}
pd.DataFrame(summ).T
```

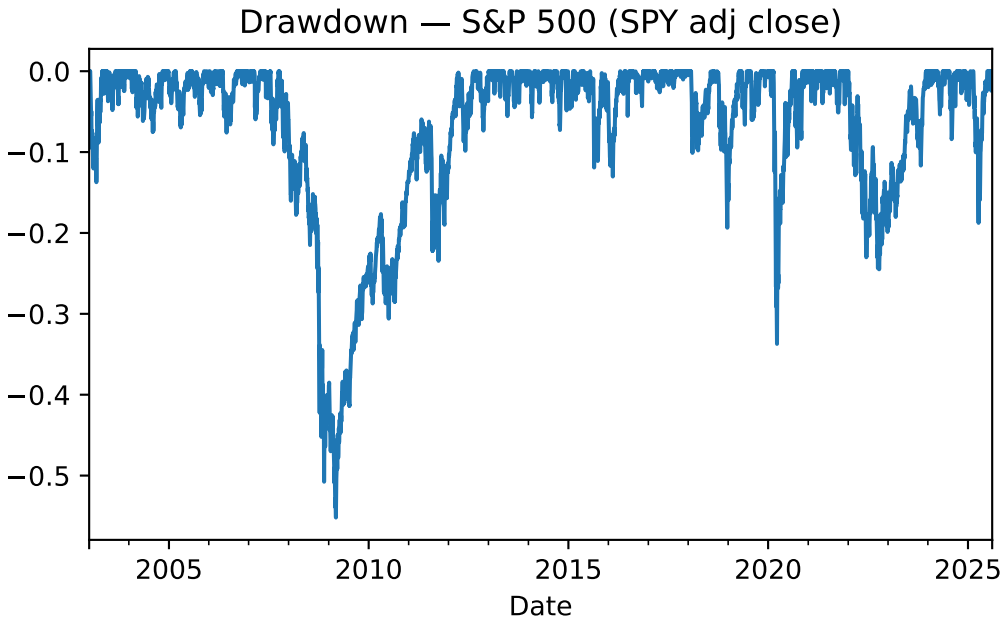
```
Ticker            SPY
Date
2003-01-02  59.986370
2003-01-03  60.170784
2003-01-06  61.231274
2003-01-07  61.079792
2003-01-08  60.197128
Ticker            IEF
Date
2003-01-02  44.324085
```

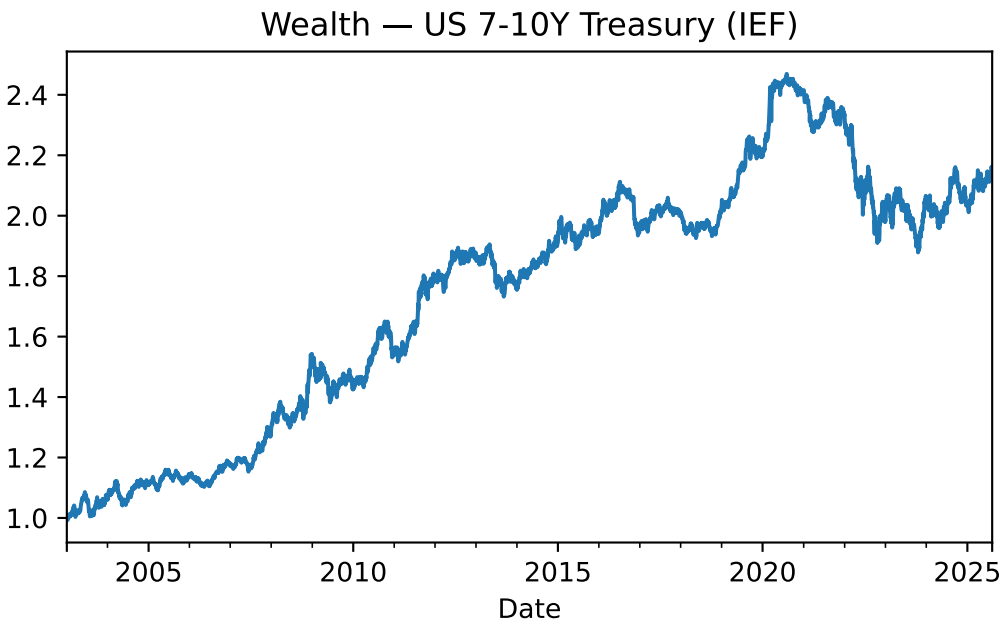
```
2003-01-03  44.397270
2003-01-06  44.287552
2003-01-07  44.428650
2003-01-08  44.533199
Ticker         BTC-USD
Date
2014-09-17  457.334015
2014-09-18  424.440002
2014-09-19  394.795990
2014-09-20  408.903992
2014-09-21  398.821014
Ticker           ^VIX
Date
2003-01-02  25.389999
2003-01-03  24.680000
2003-01-06  24.910000
2003-01-07  25.129999
2003-01-08  25.530001
```

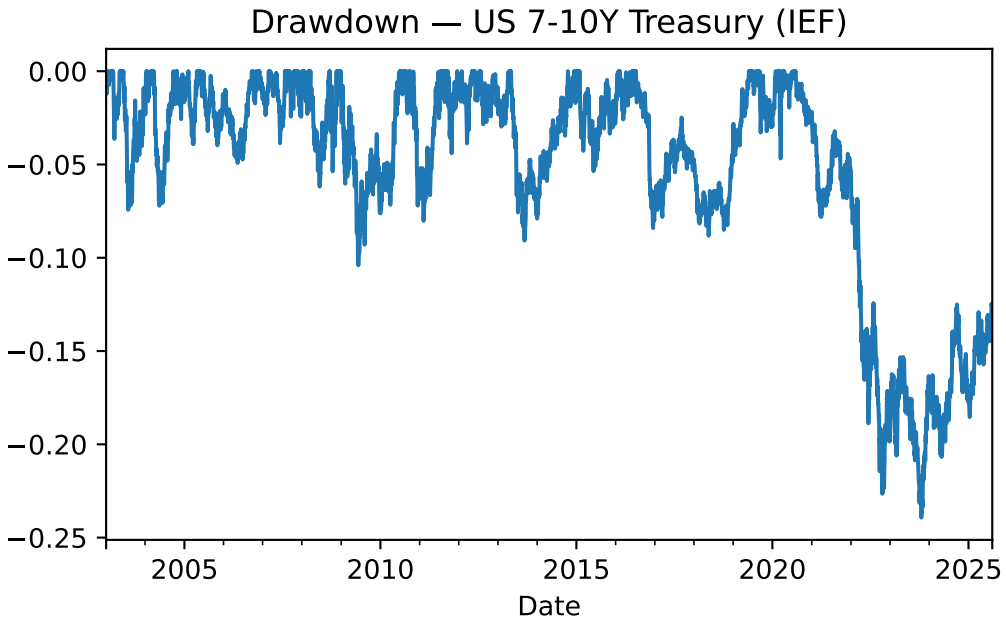
	CAGR	Vol	Sharpe	MDD	AvgDD	TUW_days
S&P 500 (SPY adj close)	0.106012	0.184301	0.575212	-0.551894	-0.078665	5175.0
US 7-10Y Treasury (IEF)	0.033232	0.067210	0.494447	-0.239248	-0.054519	5593.0
Bitcoin (BTC-USD)	0.647683	0.663650	0.975942	-0.830363	-0.370045	2695.0

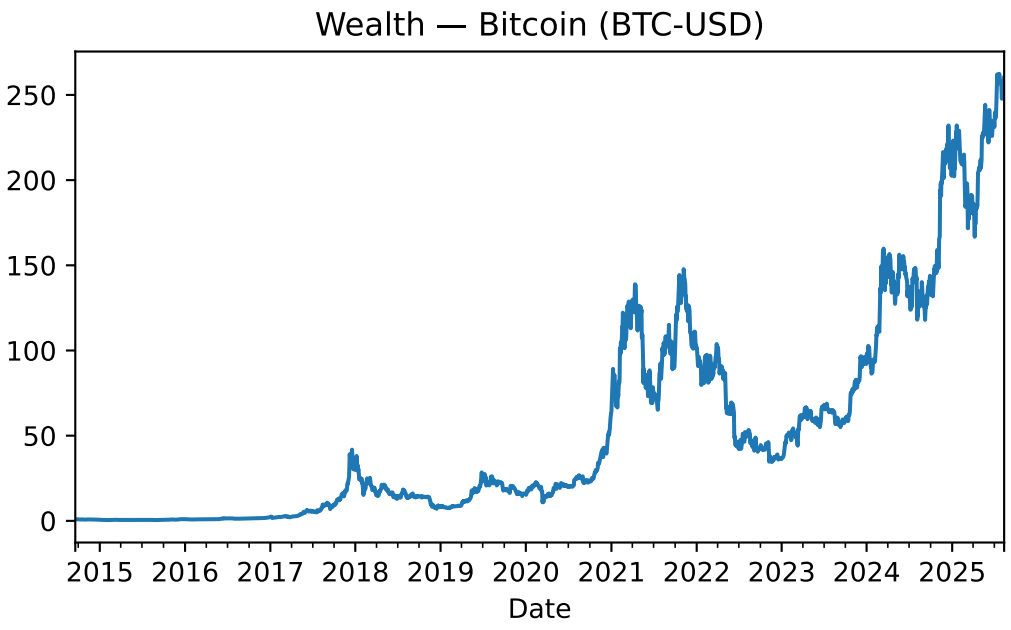
```python
# Wealth and drawdowns
for label in ["S&P 500 (SPY adj close)", "US 7-10Y Treasury (IEF)", "Bitcoin (BTC-USD)"]:
    if label not in rets:
        continue
    r = rets[label]
    W = wealth_series(r)
    dd = drawdown_series(r)
    plt.figure(); W.plot(title=f"Wealth - {label}"); plt.tight_layout(); plt.show()
    plt.figure(); dd.plot(title=f"Drawdown - {label}"); plt.tight_layout(); plt.show()
```

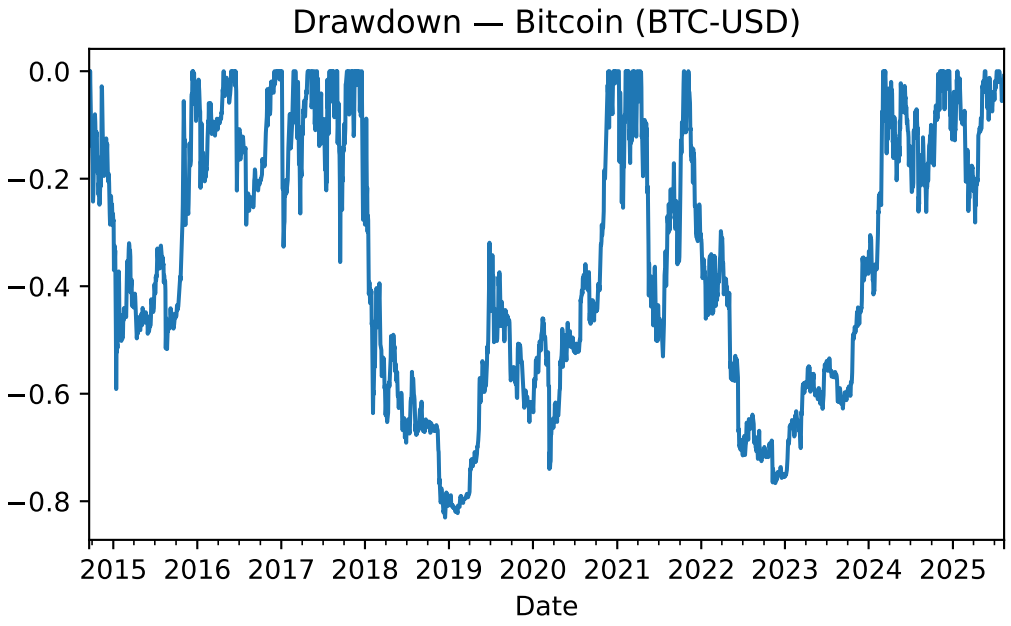

Wealth — S&P 500 (SPY adj close)
10
8
6
4
2
2005
2010
2015
2020
2025
Date


Drawdown — S&P 500 (SPY adj close)
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
2005
2010
2015
2020
2025
Date


Wealth — US 7-10Y Treasury (IEF)
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
2005
2010
2015
2020
2025
Date

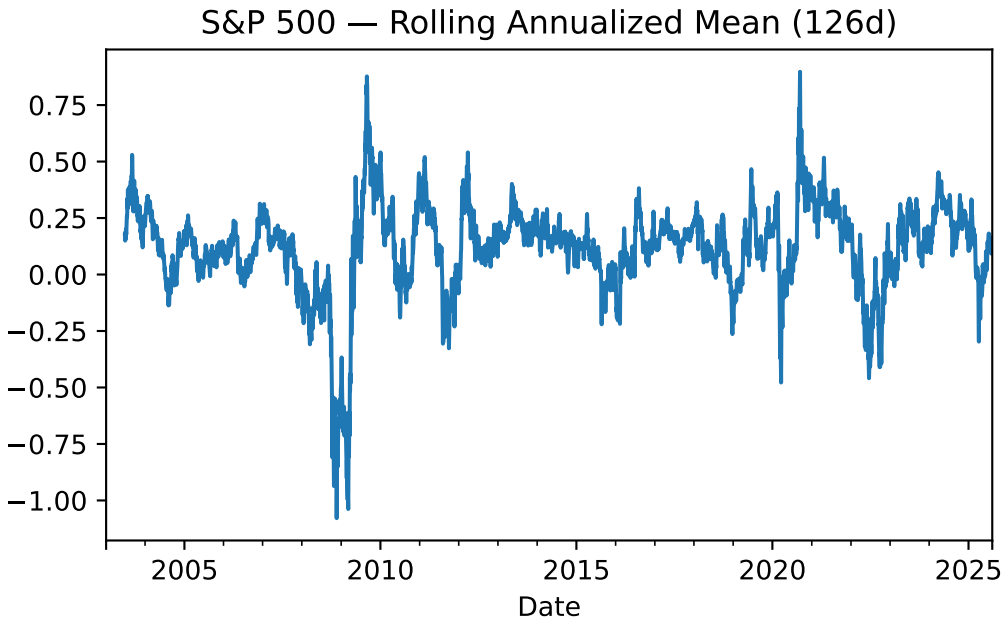

Drawdown — US 7-10Y Treasury (IEF)
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
2005
2010
2015
2020
2025
Date

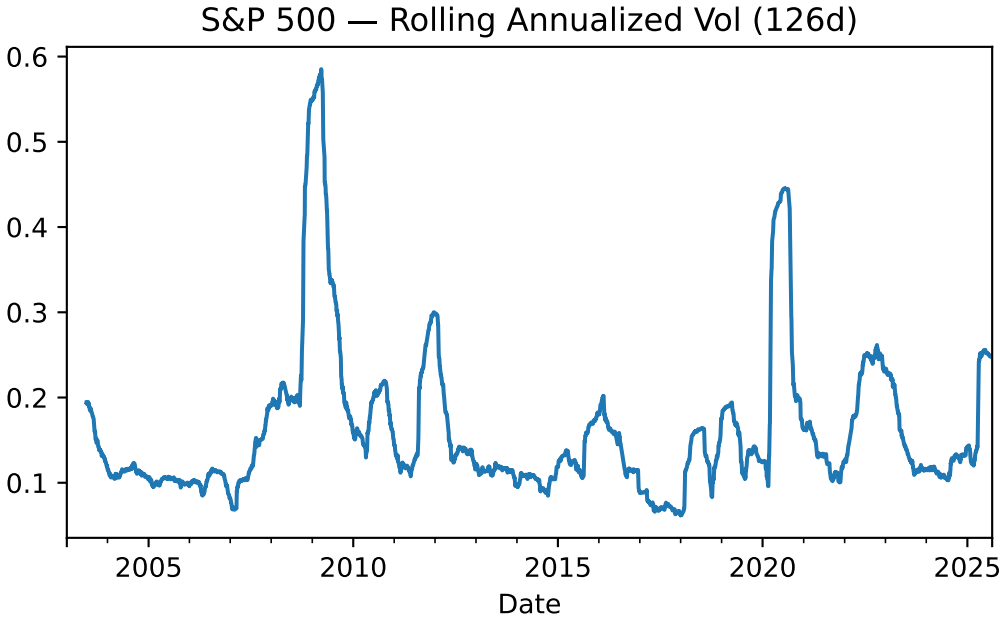


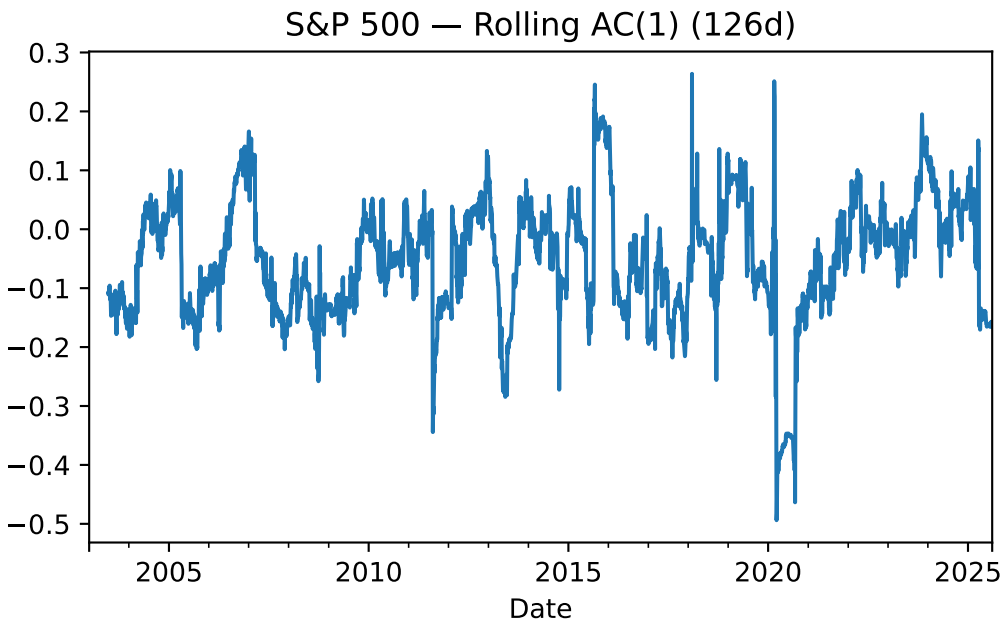



```
# Rolling stats (S&P 500 example)
label = "S&P 500 (SPY adj close)"
if label in rets:
    r = rets[label]
    roll = rolling_stats(r, w_short=20, w_long=126)
```

```
    plt.figure(); roll["mu_ann"].plot(title="S&P 500 - Rolling Annualized Mean (126d)"); plt
    plt.figure(); roll["vol_ann"].plot(title="S&P 500 - Rolling Annualized Vol (126d)"); plt
    plt.figure(); roll["rho1"].plot(title="S&P 500 - Rolling AC(1) (126d)"); plt.tight_layou
```








```
from scripts.reporting import compare_strategies, save_table
rets_named = {k: v for k, v in rets.items() if k in ["S&P 500 (SPY adj close)", "US 7-10Y Tr
if len(rets_named) >= 1:
    table = compare_strategies(rets_named)
    display(table)
    out_csv = save_table(table, "figures/empirical_summary.csv")
    print("Saved summary CSV to:", out_csv)
```

Strategy	CAGR	Vol	Sharpe	MDD	AvgDD	TUW_days
S&P 500 (SPY adj close)	0.106012	0.184301	0.575212	-0.551894	-0.078665	5175
US 7-10Y Treasury (IEF)	0.033232	0.067210	0.494447	-0.239248	-0.054519	5593
Bitcoin (BTC-USD)	0.647683	0.663650	0.975942	-0.830363	-0.370045	2695

Empirical summary exported to CSV

```
Saved summary CSV to: figures/empirical_summary.csv
```

6 Early Warning Signals

We build features from price-only data (volatility ratio, drawdown slope, SMA deviation, and a lightweight LPPL risk score) and predict whether a large drawdown occurs within a forward horizon.

```
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve, classification_report
from scripts.empirical_data import load_prices_yf
from scripts.early_warnings import build_ews_features_from_prices, label_future_drawdown_fro

spy = load_prices_yf("SPY", start="2000-01-01")
X = build_ews_features_from_prices(spy)
y = label_future_drawdown_from_prices(spy, horizon=60, threshold=-0.15)

df = pd.concat([X, y], axis=1).dropna()
split = int(len(df)*0.7)
X_train, X_test = df.iloc[:split, :-1], df.iloc[split:, :-1]
y_train, y_test = df.iloc[:split, -1], df.iloc[split:, -1]

clf = LogisticRegression(max_iter=2000)
clf.fit(X_train, y_train)
proba = clf.predict_proba(X_test)[:,1]
auc = roc_auc_score(y_test, proba)
print(f"AUC: {auc:.3f}")

# ROC
fpr, tpr, _ = roc_curve(y_test, proba)
plt.figure(); plt.plot(fpr, tpr, label=f"AUC={auc:.3f}"); plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
plt.title("ROC - EWS with LPPL (SPY)"); plt.legend(); plt.tight_layout(); plt.show()

# Feature importances (coef magnitude)
imp = pd.Series(clf.coef_[0], index=X.columns).sort_values(key=np.abs, ascending=False)
print("Top features:\n", imp.head(10))
```

```
# Classification report at 0.5 threshold
pred = (proba >= 0.5).astype(int)
print(classification_report(y_test, pred, digits=3))

# Export predictions
out = pd.DataFrame({"proba": proba, "y_test": y_test.values}, index=y_test.index)
out.to_csv("figures/ews_lppl_spy_predictions.csv")
print("Saved predictions to figures/ews_lppl_spy_predictions.csv")
```

```
Ticker            SPY
Date
2000-01-03  92.142494
2000-01-04  88.539185
2000-01-05  88.697586
2000-01-06  87.272102
2000-01-07  92.340508
AUC: 0.636
```

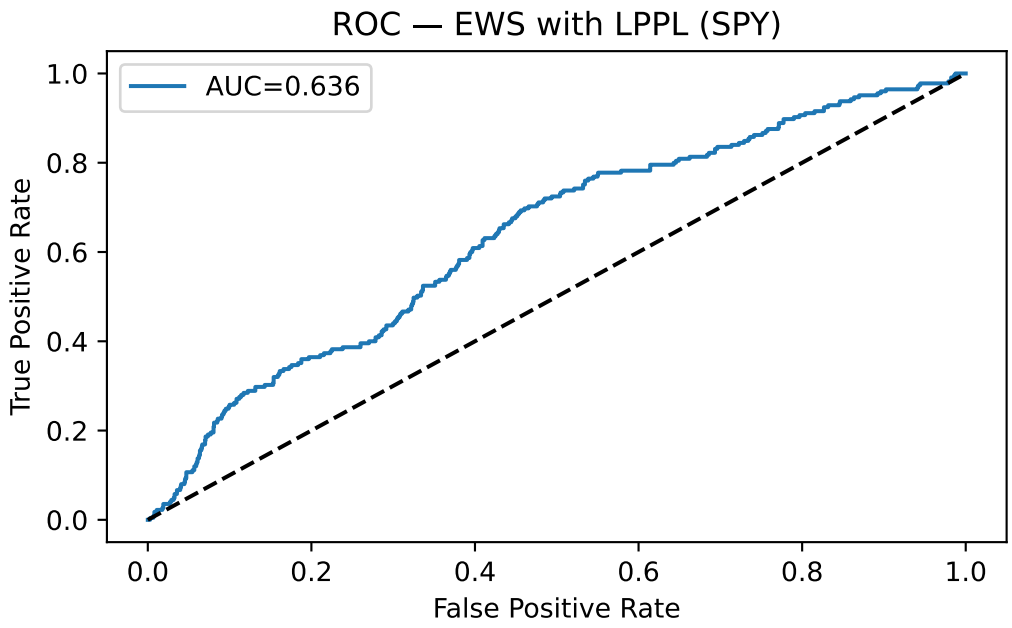


Figure 6.1: SPY early-warning model with LPPL risk

```
Top features:
 sma_dev     -10.152640
```

```
vol_ratio     0.399041
lppl_risk    -0.094796
dd_slope      0.080637
dtype: float64
              precision    recall  f1-score   support

         0.0      0.886     0.997     0.938      1742
         1.0      0.143     0.004     0.009       225

    accuracy                          0.883      1967
   macro avg      0.514     0.501     0.473      1967
weighted avg      0.801     0.883     0.832      1967

Saved predictions to figures/ews_lppl_spy_predictions.csv
```

7 Stop-Loss, Momentum, and Hedging

7.1 Demos

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from scripts.hedging import rolling_put_hedge, vix_overlay
from scripts.simulations import simulate_ar1

price = simulate_ar1(mu_ann=0.00, rho=0.2, sigma_ann=0.15, years=3, seed=123)
rets = price.pct_change().dropna()

hedged, roll_flags = rolling_put_hedge(price, moneyness=0.95, tenor_days=21, iv_annual=0.20)

wealth_asset = (1+rets).cumprod()
wealth_hedged = (1+hedged).cumprod()

plt.figure(); wealth_asset.plot(label="Asset"); wealth_hedged.plot(label="Put-hedged")
plt.title("Wealth: Asset vs Rolling 95% Put Hedge (toy)"); plt.legend(); plt.tight_layout();

vix_proxy = -rets * 3.0
trigger = rets.rolling(10).std().shift(1) > rets.rolling(60).std().shift(1)
vix_ov = vix_overlay(rets, vix_proxy, trigger, hedge_weight=0.2).dropna()
plt.figure(); (1+rets).cumprod().plot(label="Asset"); (1+vix_ov).cumprod().plot(label="VIX ov
plt.title("Wealth: Asset vs VIX Overlay (toy)"); plt.legend(); plt.tight_layout(); plt.show()
```

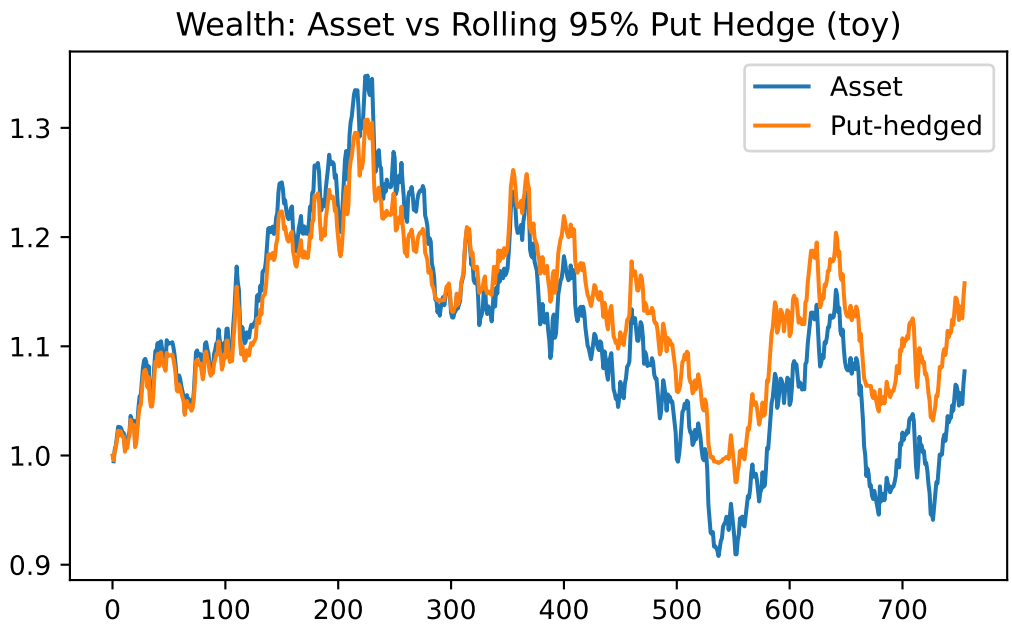
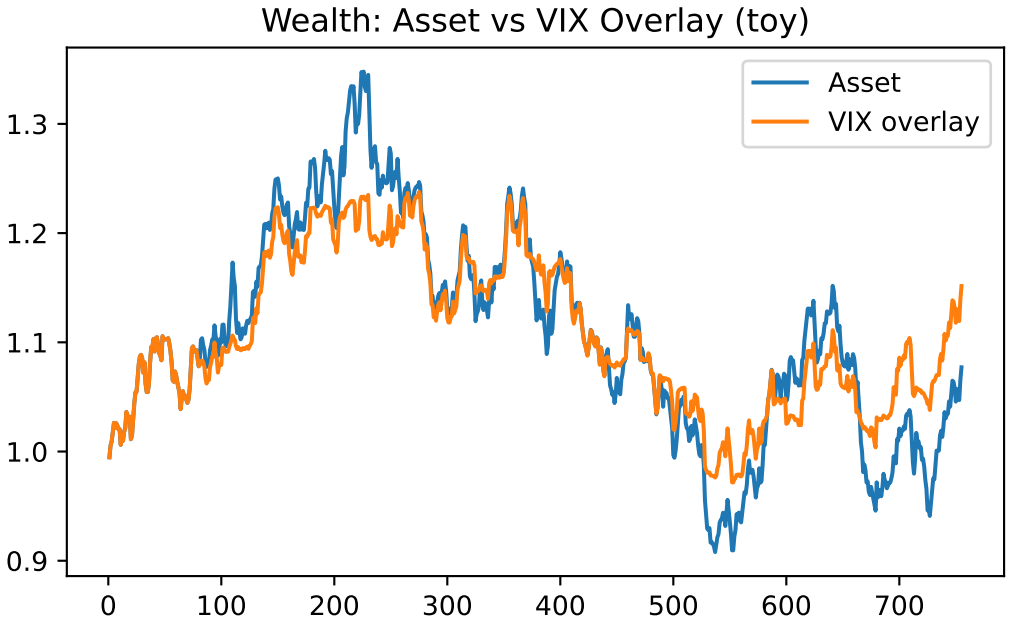


Figure 7.1: Illustrative hedging overlays (toy examples)




```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from scripts.simulations import simulate_ar1
```

```
from scripts.stoploss import apply_trailing_stop, apply_costs

price = simulate_ar1(mu_ann=0.04, rho=0.1, sigma_ann=0.15, years=5, seed=7)
rets = price.pct_change().dropna()

sl_rets, pos, trade, tovr = apply_trailing_stop(price, stop_pct=0.05, reenter_above_peak=Tru
sl_costed = apply_costs(sl_rets, trade, tovr, fixed_bps=1.0, slippage_bps=2.0, spread_bps=5.

wealth_bh = (1+rets).cumprod()
wealth_sl = (1+sl_rets).cumprod()
wealth_sl_cost = (1+sl_costed).cumprod()

plt.figure(); wealth_bh.plot(label="Buy & Hold"); wealth_sl.plot(label="Stop 5% (no costs)")
plt.title("Wealth with Stop-Loss and Transaction Costs (toy)"); plt.legend(); plt.tight_layou
```

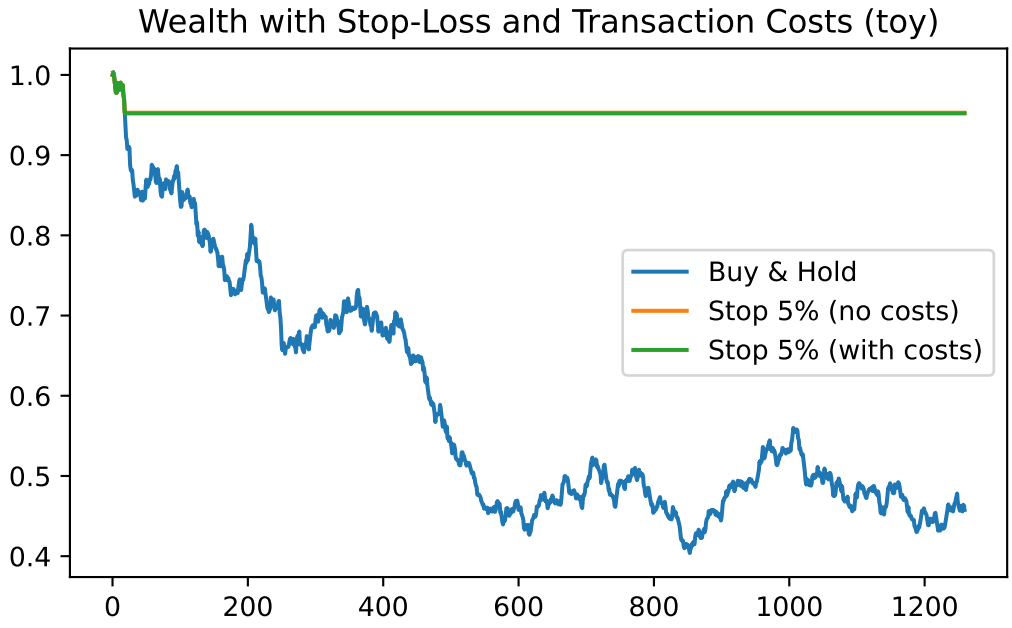


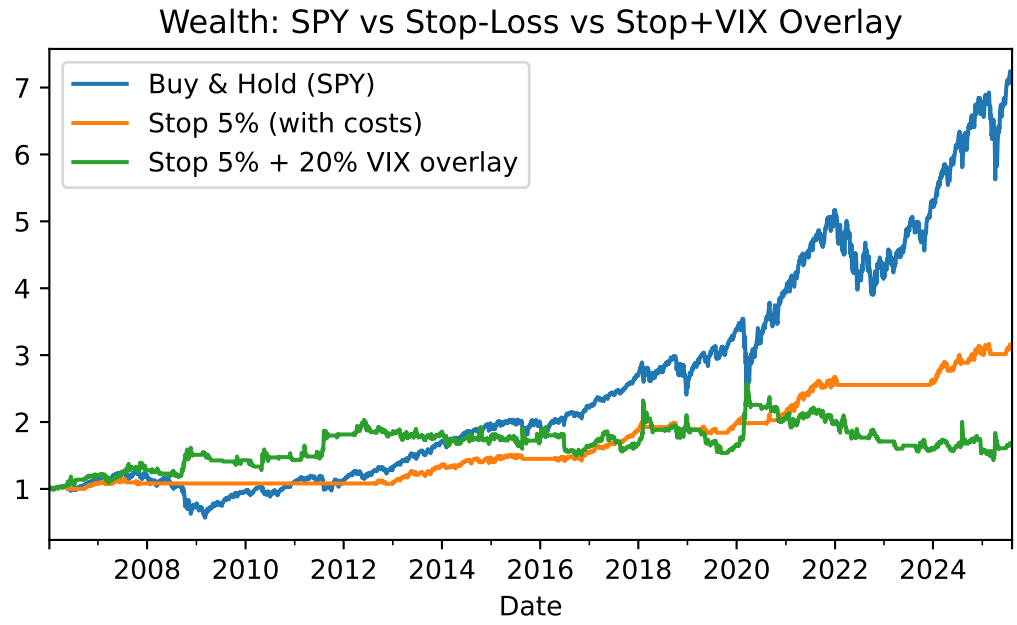
Figure 7.2: Stop-loss with transaction costs (toy example)

7.2 Data-Driven Hedge Example (VIX proxy)

```
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from scripts.empirical_data import load_prices_yf, pct_returns, wealth_series
from scripts.hedging import vix_overlay
from scripts.stoploss import apply_trailing_stop, apply_costs

spy = load_prices_yf("SPY", start="2006-01-01")
vix = load_prices_yf("^VIX", start="2006-01-01")

r_spy = pct_returns(spy)
r_vix = pct_returns(vix)

vol_s = r_spy.rolling(20).std().shift(1)
vol_l = r_spy.rolling(60).std().shift(1)
trigger = (vol_s > vol_l).reindex_like(r_spy).fillna(False)

sl_rets, pos, trade, tovr = apply_trailing_stop(spy, stop_pct=0.05, reenter_above_peak=True)
sl_rets = sl_rets.reindex_like(r_spy).fillna(0.0)
sl_costed = apply_costs(sl_rets, trade, tovr, fixed_bps=1.0, slippage_bps=2.0, spread_bps=5.0

combo = vix_overlay(sl_costed, r_vix, trigger, hedge_weight=0.20).dropna()

W_bh = wealth_series(r_spy)
W_sl = wealth_series(sl_costed)
W_combo = wealth_series(combo)

plt.figure(); W_bh.plot(label="Buy & Hold (SPY)"); W_sl.plot(label="Stop 5% (with costs)"); W
plt.title("Wealth: SPY vs Stop-Loss vs Stop+VIX Overlay"); plt.legend(); plt.tight_layout();
```

```
Ticker            SPY
Date
2006-01-03  87.964905
2006-01-04  88.381447
2006-01-05  88.437012
2006-01-06  89.172943
2006-01-09  89.402100
Ticker       ^VIX
Date
2006-01-03  11.14
2006-01-04  11.37
2006-01-05  11.31
2006-01-06  11.00
```

2006-01-09 11.13

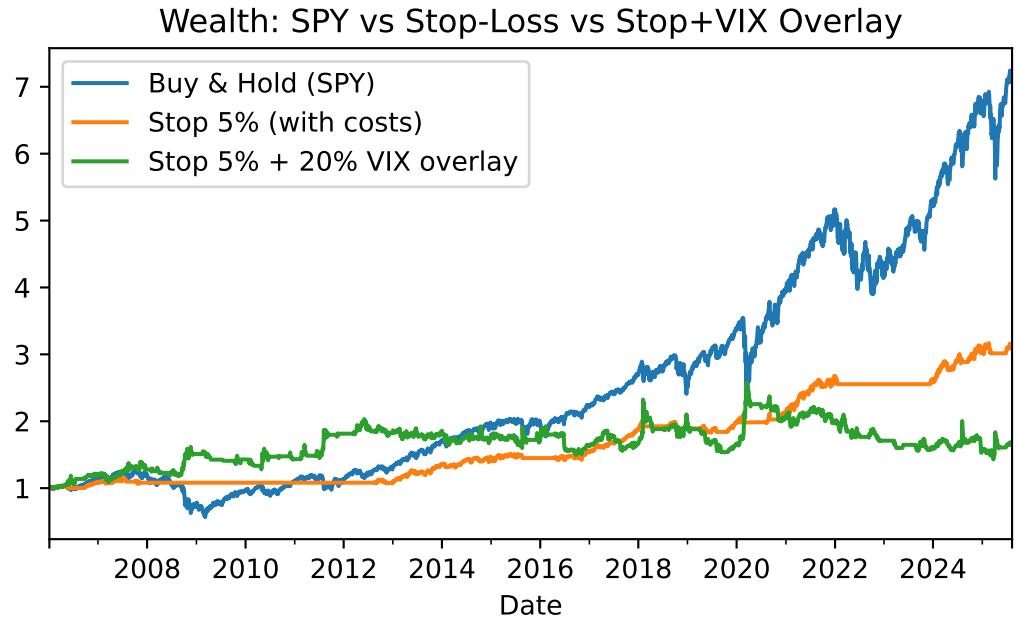



```
from scripts.reporting import compare_strategies, save_table, save_figure
from scripts.empirical_data import wealth_series

comparables = {
    "Buy & Hold (SPY)": r_spy,
    "Stop 5% (with costs)": sl_costed,
    "Stop 5% + 20% VIX overlay": combo
}
table = compare_strategies(comparables)
display(table)

csv_path = save_table(table, "figures/spy_strategy_comparison.csv")
print("Saved strategy metrics to:", csv_path)

plt.figure()
wealth_series(r_spy).plot(label="Buy & Hold (SPY)")
wealth_series(sl_costed).plot(label="Stop 5% (with costs)")
wealth_series(combo).plot(label="Stop 5% + 20% VIX overlay")
plt.title("Wealth: SPY vs Stop-Loss vs Stop+VIX Overlay"); plt.legend()
png_path = save_figure("figures/spy_wealth_strategies.png")
print("Saved figure to:", png_path)
```

Strategy	CAGR	Vol	Sharpe	MDD	AvgDD	TUW_days
Buy & Hold (SPY)	0.102128	0.191638	0.532922	-0.551895	-0.086434	4495
Stop 5% (with costs)	0.058013	0.072148	0.804087	-0.087379	-0.039348	4739
Stop 5% + 20% VIX overlay	0.025733	0.179518	0.143343	-0.445388	-0.168309	5034

SPY: Buy&Hold vs Stop vs Stop+VIX — metrics and exports

```
Saved strategy metrics to: figures/spy_strategy_comparison.csv
Saved figure to: figures/spy_wealth_strategies.png
```



8 Results Synthesis

(Write-up and comparison tables to be added.)

9 Appendices

(Math notes, parameter grids, and full tables to be added.)

Chekhlov, Alex, Stanislav Uryasev, and Roman Zabarankin. 2005. "Drawdown Measure in Portfolio Optimization." In *International Conference on Computational Science*, 294–301. Springer. https://doi.org/10.1007/11428831_35.

Conlon, Thomas, Heather J. Ruskin, and Martin Crane. 2010. "Cross-Correlation Dynamics in Financial Markets." *Physica A: Statistical Mechanics and Its Applications* 389 (21): 5228–39. https://doi.org/10.1016/j.physa.2010.05.050.

Filimonov, Vladimir, and Didier Sornette. 2013. "Apparent Criticality and Calibration Issues in the Log-Periodic Power Law Model." *Quantitative Finance* 13 (11): 1701–20. https://doi.org/10.1080/14697688.2013.802144.

Goldberg, Lisa R., and Ola Mahmoud. 2014. "Drawdown: From Practice to Theory and Back Again." *Mathematics and Financial Economics* 8 (3): 211–49. https://doi.org/10.1007/s11579-014-0119-8.

Hadjiliadis, Olympia, and Jan Vecer. 2006. "Drawdowns Preceding Rallies in Diffusion Models." *Quantitative Finance* 6 (5): 403–9. https://doi.org/10.1080/14697680600815604.

Johansen, Anders, Olivier Ledoit, and Didier Sornette. 2000. "Crashes as Critical Points." *International Journal of Theoretical and Applied Finance* 3 (2): 219–55. https://doi.org/10.1142/S0219024900000115.

Kaminski, Kathryn M., and Andrew W. Lo. 2014. "When Do Stop-Loss Rules Stop Losses?" *Journal of Portfolio Management* 40 (2): 45–61. https://doi.org/10.3905/jpm.2014.40.2.045.

Lo, Andrew W., and Alexander Remorov. 2017. "Stop-loss Strategies with Serial Correlation, Regime Switching, and Transaction Costs." *Journal of Financial Markets* 34: 1–15. https://doi.org/10.1016/j.finmar.2017.02.003.

Magdon-Ismail, Malik, Amir F. Atiya, Amit Pratap, and Yaser S. Abu-Mostafa. 2004. "Maximum Drawdown for Brownian Motion." *Journal of Applied Probability* 41 (1): 147–61. https://doi.org/10.1239/jap/1077134663.

Sornette, Didier. 2003. *Why Stock Markets Crash: Critical Events in Complex Financial Systems*. Princeton, NJ: Princeton University Press.