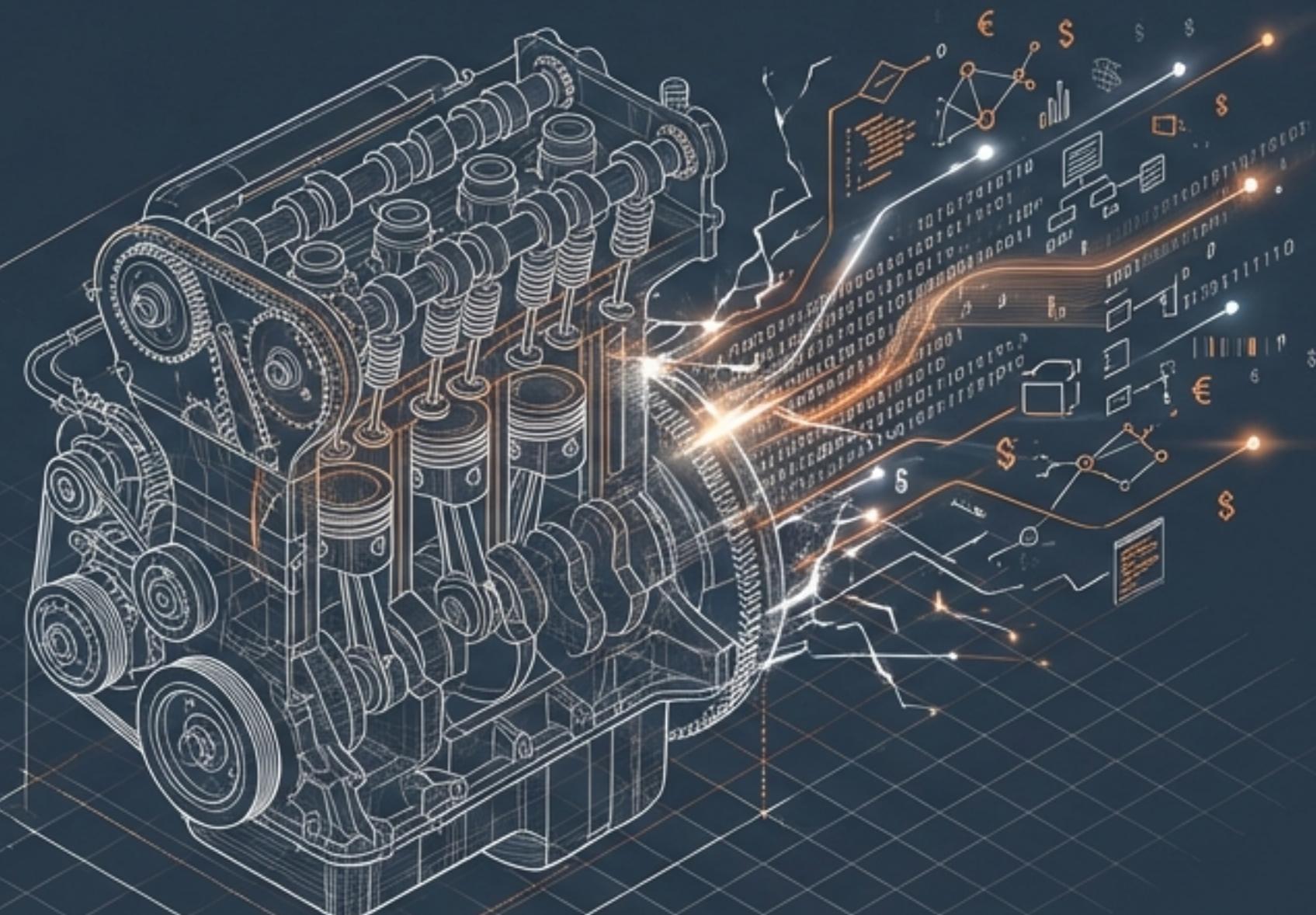


백테스팅과 자동매매 시스템: 퀀트 트레이딩을 위한 생존 가이드

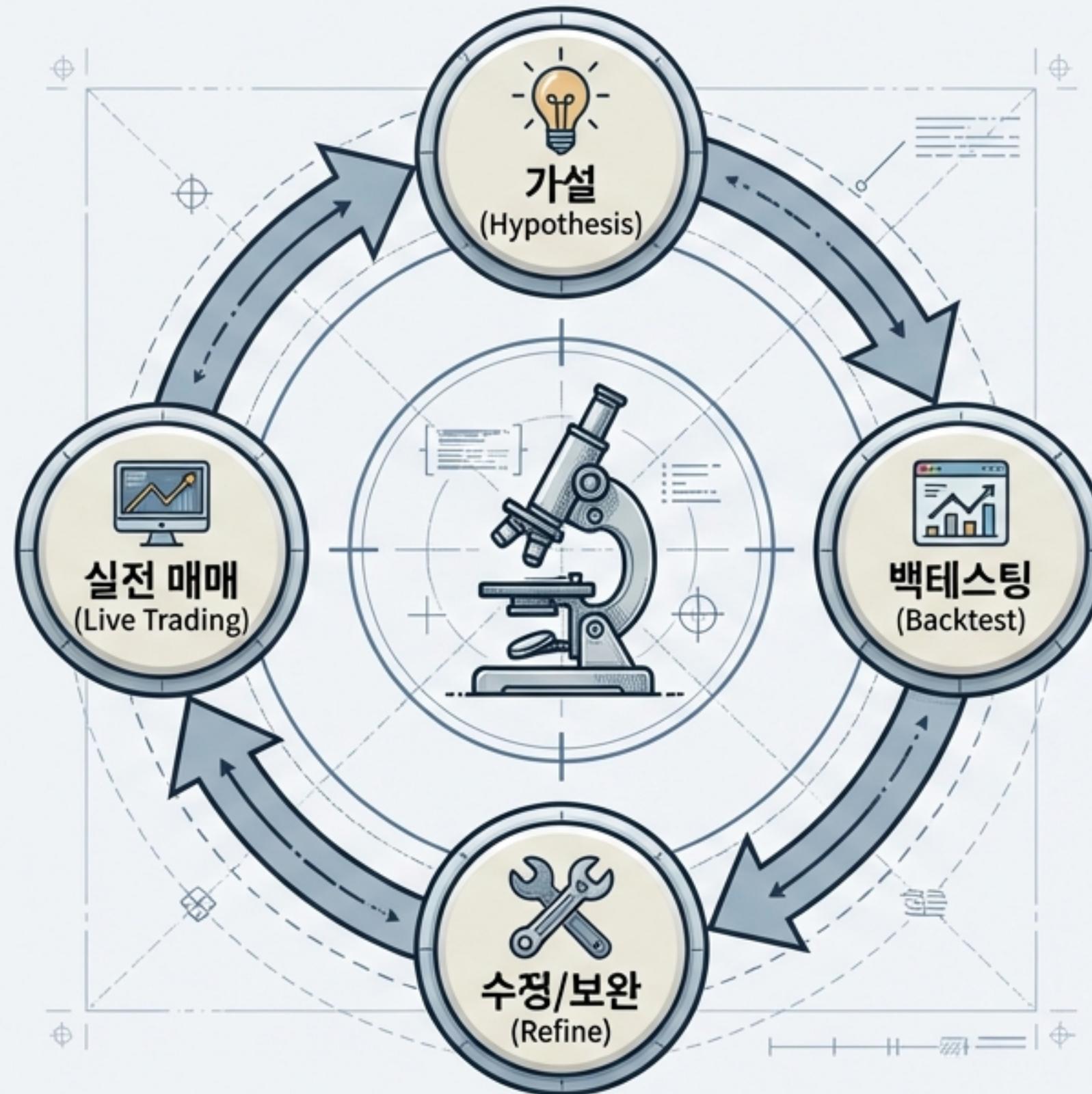
Backtesting and Automated Execution: A Survival Guide



알고리즘 아키텍트를 위한 현장 매뉴얼

수익률 추정은 시작일 뿐입니다. 백테스팅의 진정한 목적은 구현의 세부 사항을 디버깅하고, 실제 시장의 가혹한 현실에서 살아남을 수 있는 견고한 시스템을 설계하는 것입니다.

백테스팅의 진정한 목적: 수익률 그 이상의 검증

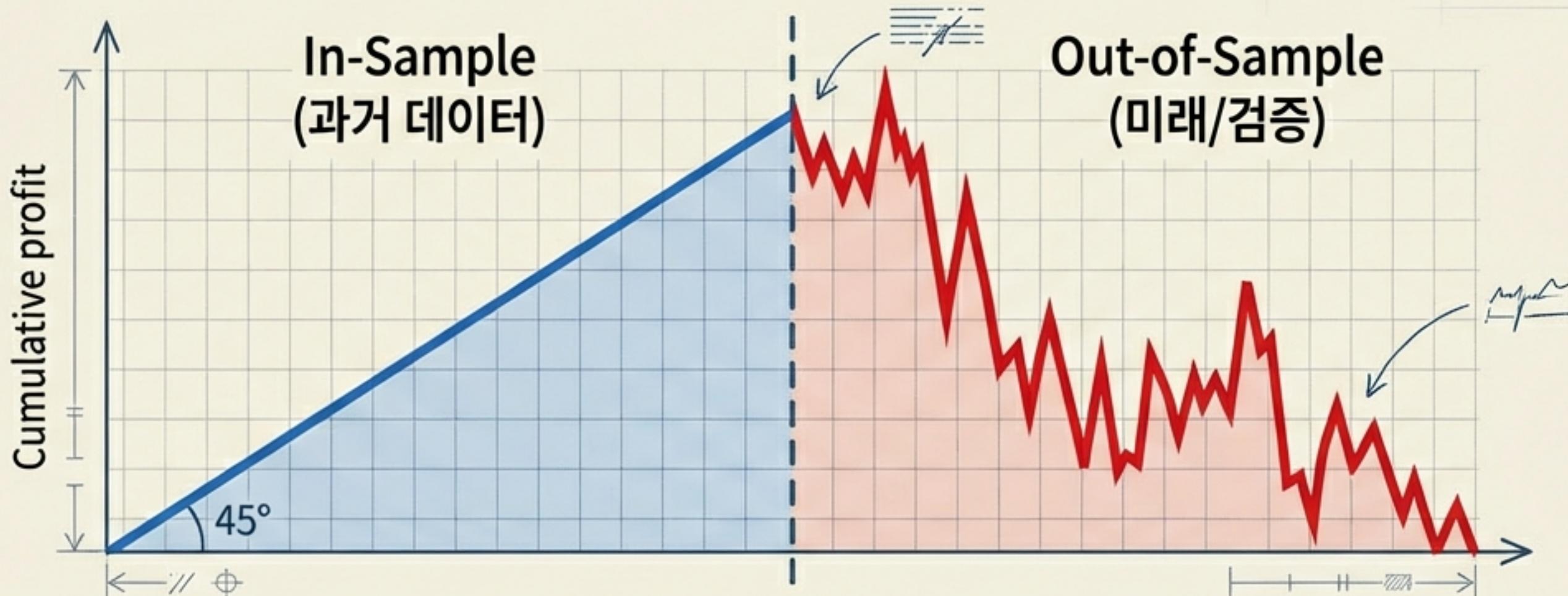


구현의 디테일 (The Devil is in the Details)

- 전략의 수익성은 미세한 구현 차이에서 결정됩니다.
- MOO (시가) vs. 장 시작 직후 시장가 주문
- 오후 4:00 (주식 마감) vs. 오후 4:15 (선물 마감)

게재된 전략을 그대로 믿지 마십시오.
직접 백테스팅하여 입찰가(Bid)를 쓸지,
매도호가(Ask)를 쓸지, 혹은 체결가(Last
Price)를 쓸지 결정해야 합니다.

논리적 오류: 코드가 당신을 속일 때



미래 참조 편향 (Look-ahead Bias)

내일의 데이터(고가/저가)를
오늘 거래에 사용하는 오류.
백테스팅과 실전 코드가 다를 때 발생.

데이터 스누핑 (Data-Snooping)

과거 데이터의 노이즈에
과적합(Over-fitting)된 상태.
우연을 패턴으로 착각함.

선형 모델의 미학 (Occam's Razor)

단순한 모델이 최고입니다.
비선형 모델은 과거엔 완벽하지만
미래 예측력은 떨어집니다.

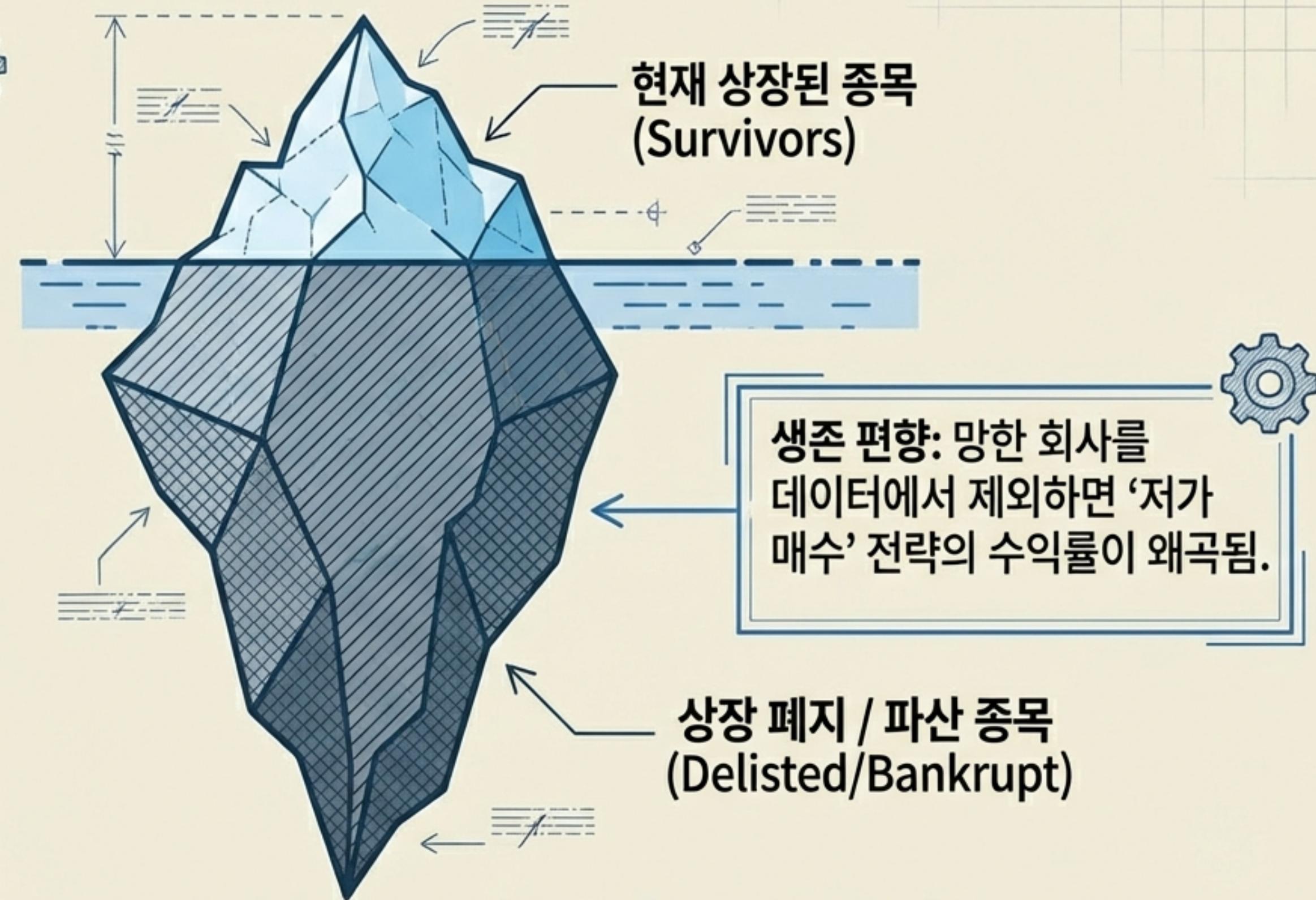
데이터 위생: 수정주가와 생존 편향의 함정



액면분할 (Splits):
N 대 1 분할 시 가격을
N으로 나누었는가?

배당락 (Dividends):
현금 배당만큼 과거 주가를
하향 조정했는가?

데이터 소스: 편향 없는
데이터(CSI, Kibot)를
사용하고 있는가?

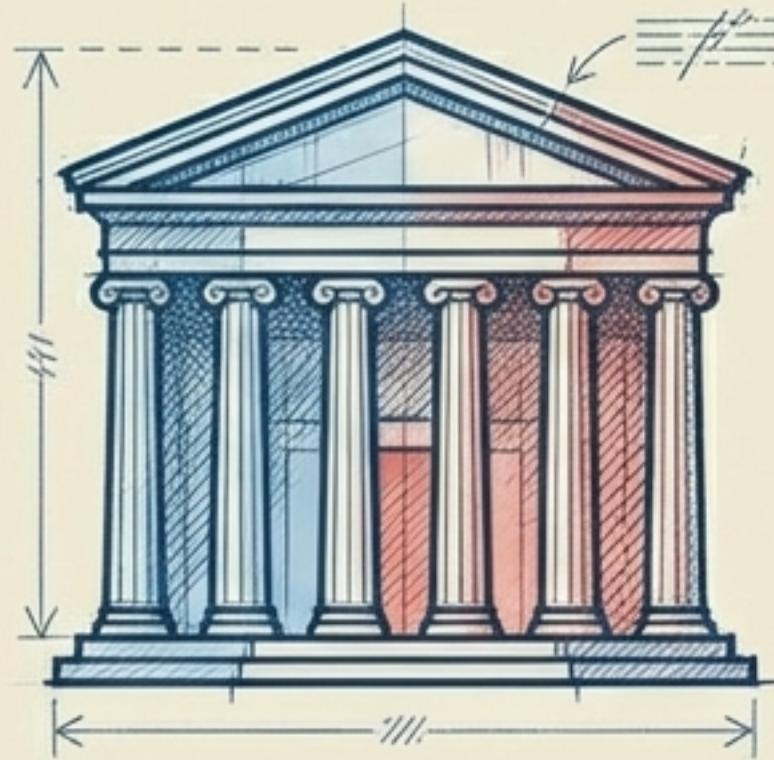


생존 편향: 망한 회사를
데이터에서 제외하면 ‘저가
매수’ 전략의 수익률이 왜곡됨.

**상장 폐지 / 파산 종목
(Delisted/Bankrupt)**

시장 미시구조: 가격과 유동성의 괴리

주식 시장의 파편화



통합 시세(Consolidated)와
주거래소(Primary)의 차이.
MOC(종가) 주문은 오직
주거래소에서만 체결됨.
보조 거래소의 가격 왜곡 주의.

FX 시장의 특수성



중앙 거래소가 없음. 브로커마다
호가와 스프레드가 다름.
거래량 데이터가 부정확하므로
Bid/Ask 호가 데이터 사용 필수.

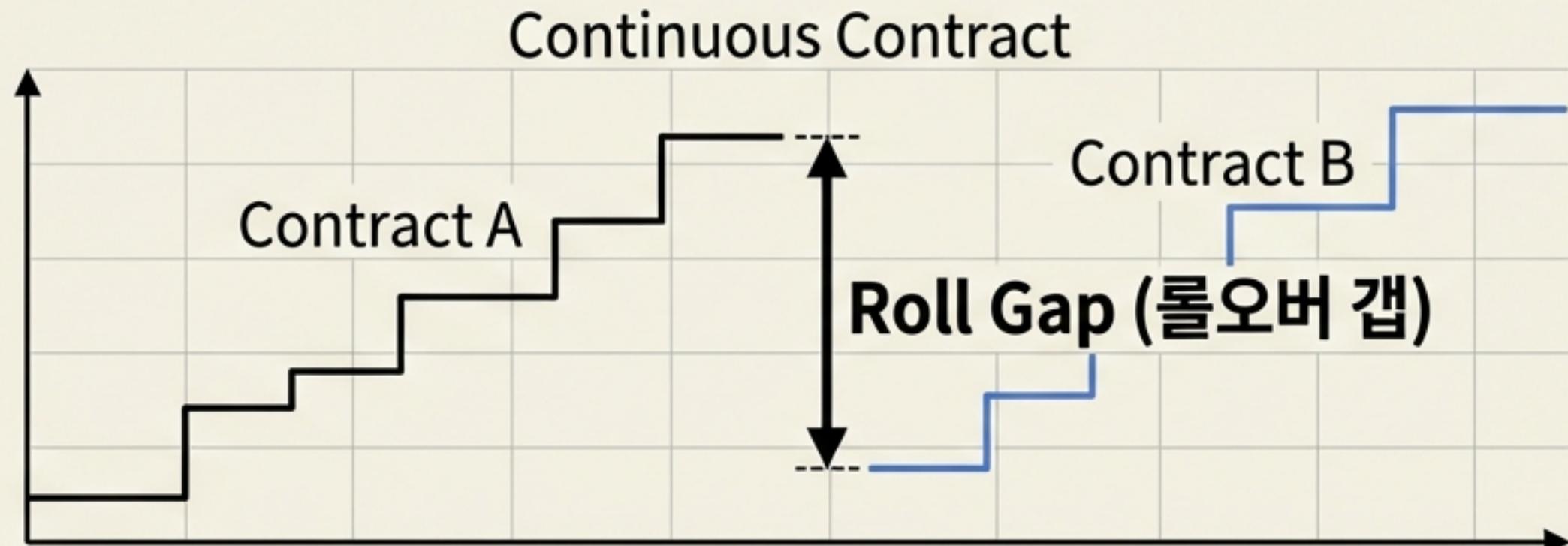
공매도 제약 (Short-Selling)



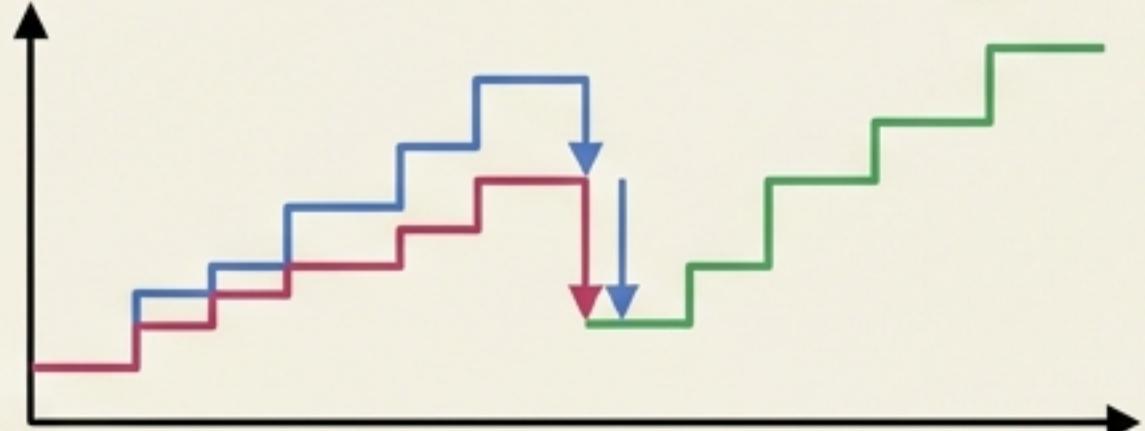
대차 불가능(Hard-to-Borrow)
종목 확인.
2008년 금융주 공매도 금지와
업틱룰(Uptick Rule) 같은
규제 사항 반영 필수.



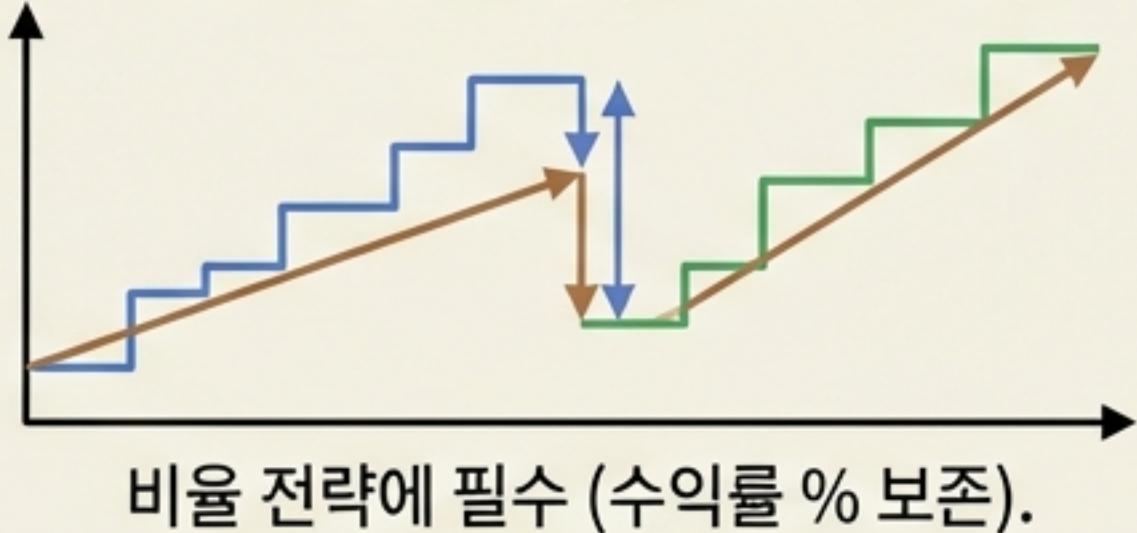
선물 거래: 만기일과 롤오버의 수학



가격 보정 (Price Back-Adjustment)
스프레드 전략에 필수 (P&L 보존).



수익률 보정 (Return Back-Adjustment)
비율 전략에 필수 (수익률 % 보존).



데이터 기준

- 일봉 데이터: 정산가 (Settlement Price) 사용
- 장중 데이터: 체결가 (Trade Price) 사용
- 타이밍 불일치 주의:
금 선물(1:30 PM) vs
골드 ETF(4:00 PM)

통계적 유의성 검증: 실력인가, 운인가?



가우시안 가정 (Sharpe Ratio)

수익률이 정규분포를
따른다고 가정.
($\text{Sharpe} \times \sqrt{n} \geq 2.326$)

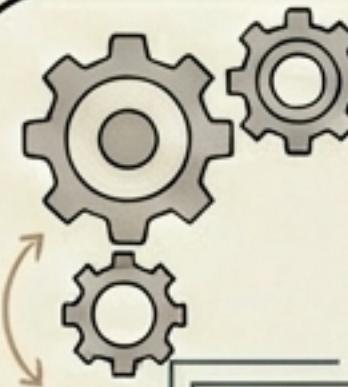
몬테카를로 시뮬레이션

수천 개의 가상 가격
시나리오 생성 후 전략
테스트.

거래 시점 무작위화 (Lo-Mamaysky-Wang)

실제 가격 데이터에서
진입 시점만 섞어서 테스트.
운에 의한 수익인지 검증.

백테스팅을 중단해야 할 때: 위험 신호



High Return / Low Sharpe

수익률은 높지만 변동성이 큼. 일관성이 부족하며 운일 가능성이 높음.



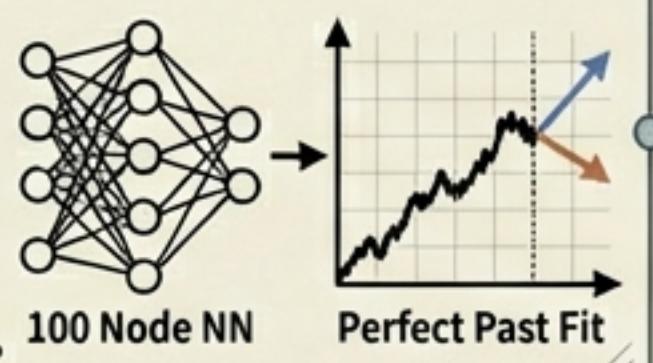
감당 불가능한 MDD

2년 이상의 낙폭 과대 구간(Drawdown). 고객이 견딜 수 없는 리스크.



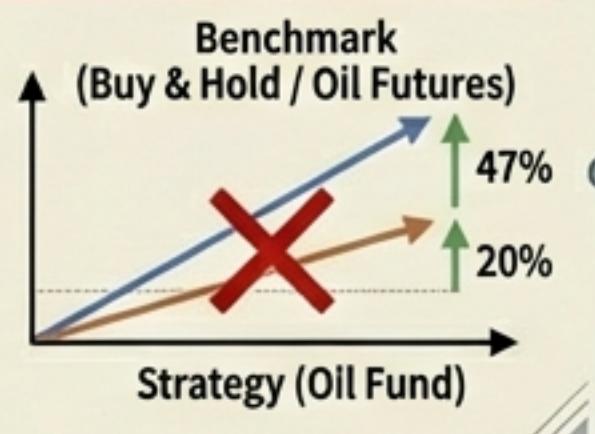
과적합 (Over-fitting)

파라미터가 너무 많은 복잡한 모델 (예: 100 노드 신경망). 과거만 완벽하게 설명함.

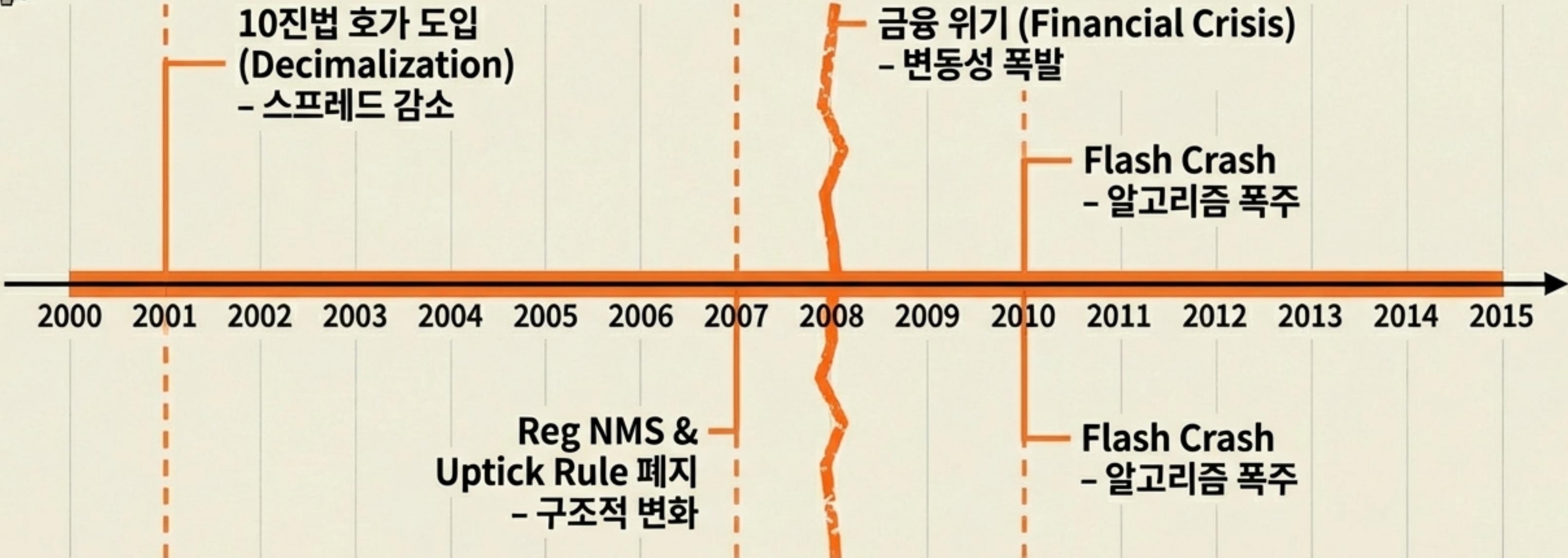


잘못된 벤치마크

단순 매수 후 보유(Buy & Hold)보다 못한 전략. (예: 2007년 오일 펀드 20% vs 오일 선물 47%)

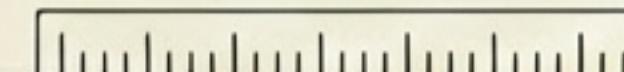


시장 체제의 변화: 과거는 미래를 보장하지 않는다



어떤 전략은 특정 체제(Regime)에서만 작동합니다.

2008년 이전에 통했던 전략이 지금도 유효하다는 보장은 없습니다.



플랫폼 선택 전략: 도구와 기술의 일치

Ease of Use

Power/Flexibility

Beginner

No-Code / GUI
(Deltix, Apama)

직관적이지만 유연성 부족

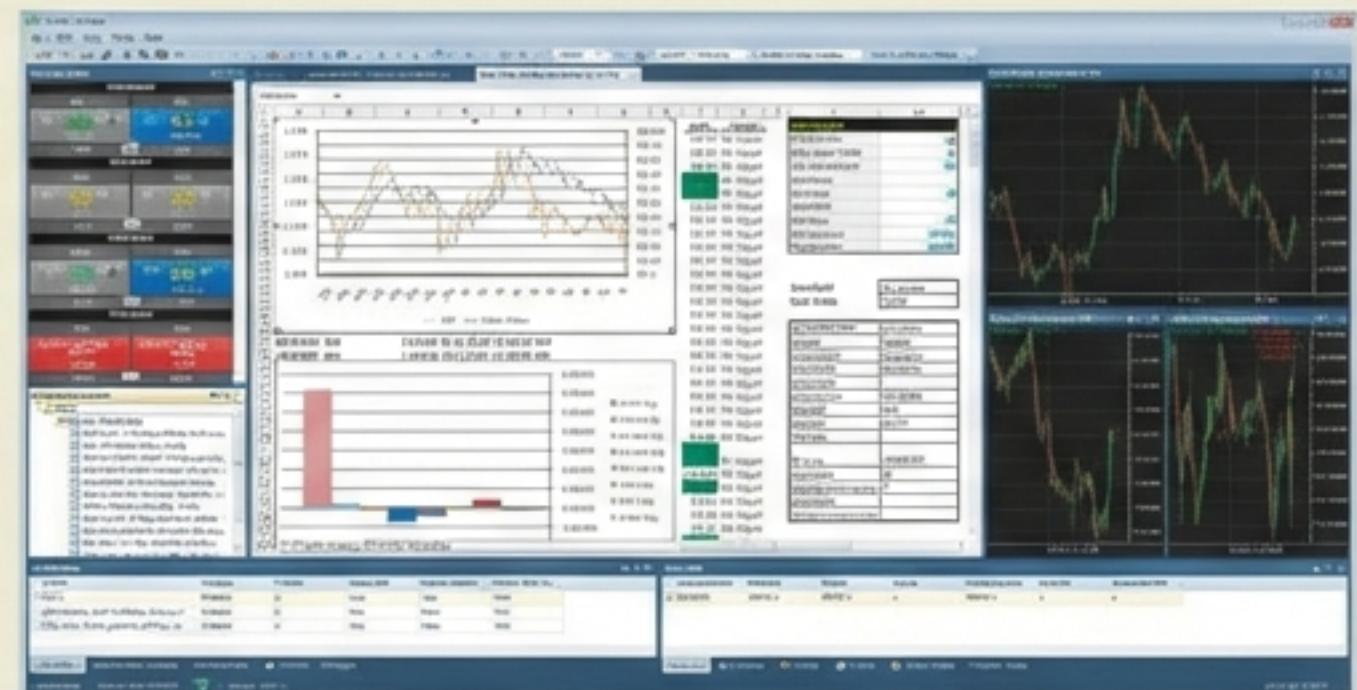
Quant/Researcher

Scripting
(MATLAB/Python/R)
연구에 최적화, 연결성 필요

Developer

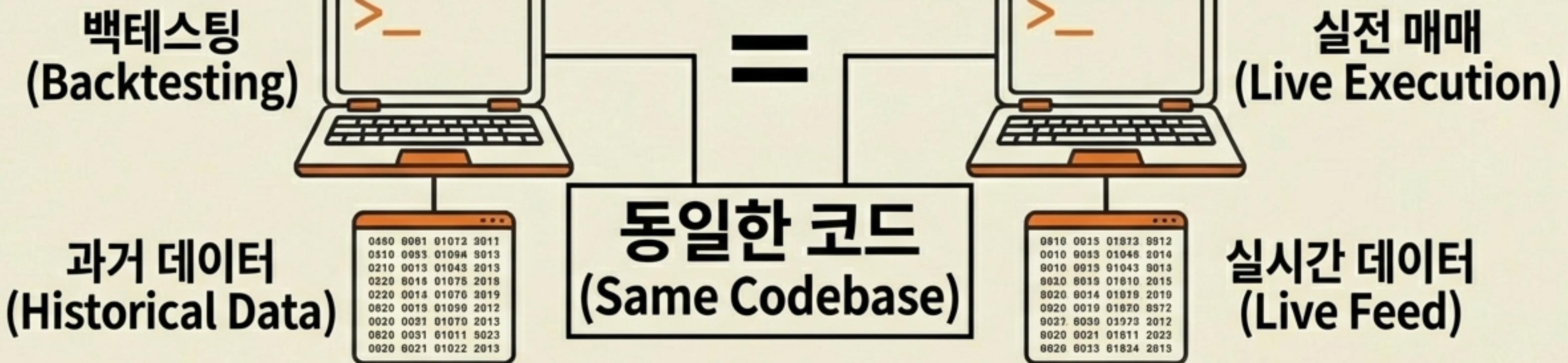
Hardcore
(C++/Java/C#)

최고의 속도와 자유도, 높은 진입장벽



예시: 엑셀과 유사하지만 강력한 C++ 엔진을 탑재한 FXone 플랫폼.

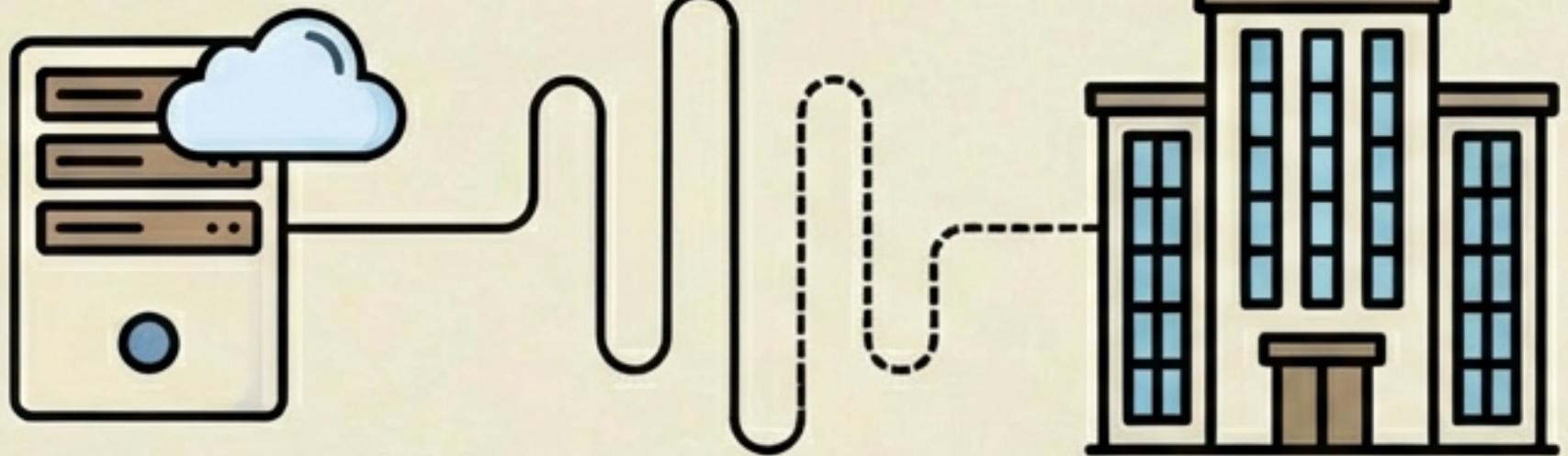
통합 아키텍처: 백테스팅과 실행의 일치



- 1. **변환 오류 제거 (No Transcription Errors)**: 언어 변환 중 발생하는 버그 방지.
- 2. **미래 참조 방지 (No Look-ahead Bias)**: 틱 단위(Tick-by-tick)로 과거를 재생하여 검증.
- 3. **이벤트 기반 시뮬레이션 (Event-Driven)**: 실제 시장과 동일한 이벤트 처리 로직.

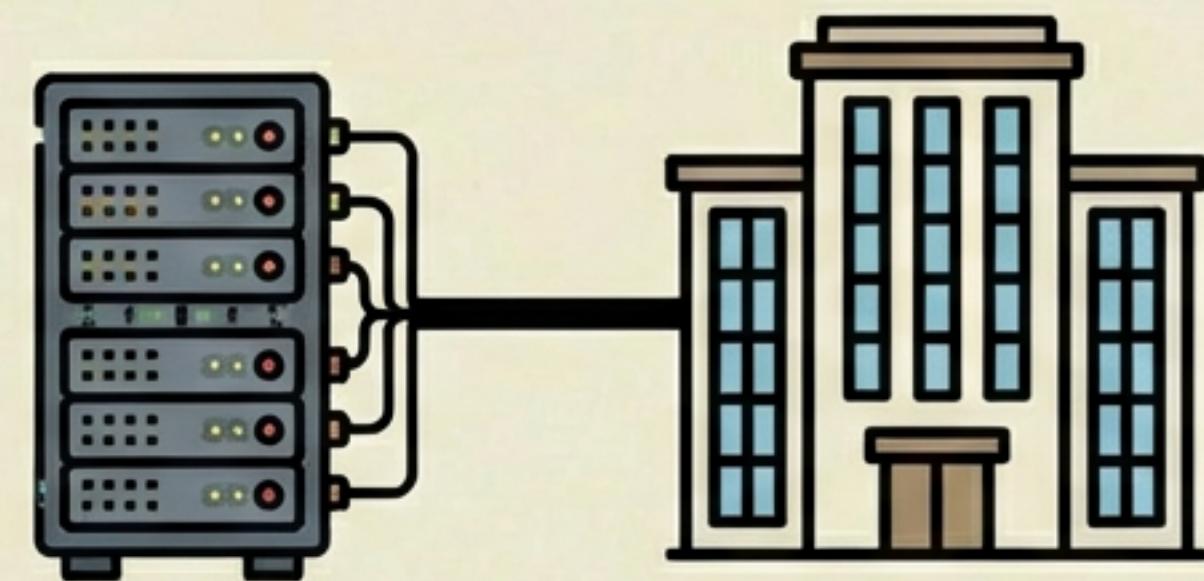
고빈도 매매의 물리적 조건: 속도와의 전쟁

Cloud / VPS



Public Internet
(Slow)

Colocation (코로케이션)



Exchange
(거래소)

Direct Fiber
(Microseconds)

- 데이터 피드: 통합 피드(SIP) 대신 거래소 직접 수신(Direct Feed) 필수.
- 동시성 처리 (Concurrency): 500개 종목 동시 거래를 위한 멀티스레딩(Multithreading) 지원 언어(C++, Java) 필요.



최종 점검 리스트: 퀸트의 생존 배낭

- [] 배당 및 분할 (Splits/Divs): 역사적 데이터가 역방향 수정(Back-adjusted) 되었는가?
- [] 생존 편향 (Survivorship): 상장 폐지된 종목이 데이터에 포함되었는가?
- [] 미래 참조 (Look-ahead): 당일의 고가/저가를 당일 진입 신호로 쓰지 않았는가?
- [] 시장 구분 (Venues): MOC 주문 시 통합 데이터가 아닌 주거래소 데이터를 썼는가?
- [] 공매도 (Shorting): 공매도 금지 기간이나 틱 규칙(Uptick Rule)을 반영했는가?
- [] 선물 롤오버 (Futures Roll): 스프레드 전략에 가격 보정, 비율 전략에 수익률 보정을 했는가?
- [] 레이턴시 (Latency): 실제 체결 지연 시간과 슬리피지를 반영했는가?

예측력의 본질 (The Essence of Predictive Power)

백테스팅은 단순히 과거의 수익을 계산하는 것이 아닙니다. 그것은 당신의 알고리즘이 미래의 불 확실성을 견딜 수 있는지 확인하는 과정입니다.

알고리즘 트레이딩은 코드만의 문제가 아닙니다. 시장의 경제적, 구조적 현실을 이해하는 자만이 '데이터 스누핑'의 함정을 넘어 진정한 '예측력'을 확보할 수 있습니다.



End of Field Guide.