Day 1: Pricing Digital Assets

Models That Actually Work

Day 1 of 5

Prof. Jörg Osterrieder

MSc Seminar – Digital Finance

Spring 2026

MSc Seminar: Digital Finance

MSc Seminar: Five Days of Digital Finance

D1 **Pricing Digital Assets** – GARCH, jump-diffusion, implied vol

D2 **DeFi Protocol Design** – AMMs, concentrated liquidity, lending

D3 **Blockchain Economics** – consensus, MEV, fee markets

D4 **AI & Crypto Markets** – ML for returns, NLP for sentiment

D5 **Risk & Regulation** – portfolio VaR, stablecoins, MiCA

Three-Level Series

- **BSc:** intuitive, graphical, minimal math
- **MSc:** intermediate math, simulation, estimation (you are here)
- **PhD:** measure theory, SDEs, proofs

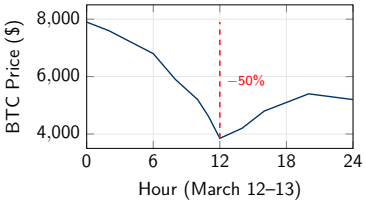
Prerequisites

Regression, MLE basics, calculus, portfolio theory, Python

What the Models Missed: March 2020 Crypto Crash

March 12–13, 2020:

- BTC fell from ~$7,900 to ~$4,800 on major exchanges (BitMEX saw a wick to $3,850)
- $\approx -40\%$ **in 24 hours**
- ETH dropped -43% same day
- $1.6B in futures liquidated
- BitMEX went offline under load



Context:

- COVID-19 panic hit all markets
- S&P 500 lost 9.5% (worst since 1987)
- But BTC fell *five times more*
- "Digital gold" narrative shattered
- Cascading liquidations amplified the move

The Question

Standard pricing models assign essentially *zero* probability to this event. Why?

An 8σ Event Under the Normal Distribution

Setup: BTC annualized volatility $\sigma \approx 70\%$, so daily $\sigma_d = 70\%/\sqrt{365} \approx 3.66\%$.

A -50% log-return: $r = \ln(0.5) = -69.3\%$ in one day.

$$z = \frac{r}{\sigma_d} = \frac{-0.693}{0.0366} \approx -18.9$$

An 8σ Event Under the Normal Distribution

Setup: BTC annualized volatility $\sigma \approx 70\%$, so daily $\sigma_d = 70\%/\sqrt{365} \approx 3.66\%$.

A -50% log-return: $r = \ln(0.5) = -69.3\%$ in one day.

$$z = \frac{r}{\sigma_d} = \frac{-0.693}{0.0366} \approx -18.9$$

Even using a generous $\sigma_d = 8\%$ (crisis-period realized vol):

$$z = \frac{-0.693}{0.08} \approx -8.7 \quad \Longrightarrow \quad \mathbb{P}(Z < -8.7) \approx 10^{-18}$$

An 8σ Event Under the Normal Distribution

Setup: BTC annualized volatility $\sigma \approx 70\%$, so daily $\sigma_d = 70\%/\sqrt{365} \approx 3.66\%$.

A -50% log-return: $r = \ln(0.5) = -69.3\%$ in one day.

$$z = \frac{r}{\sigma_d} = \frac{-0.693}{0.0366} \approx -18.9$$

Even using a generous $\sigma_d = 8\%$ (crisis-period realized vol):

$$z = \frac{-0.693}{0.08} \approx -8.7 \quad \Longrightarrow \quad \mathbb{P}(Z < -8.7) \approx 10^{-18}$$

Sigma level	**Probability**	**Expected once every...**
3σ	1.3×10^{-3}	2.8 years of daily data
5σ	2.9×10^{-7}	13,000 years
8σ	6.2×10^{-16}	$\sim 10^{12}$ years

Conclusion

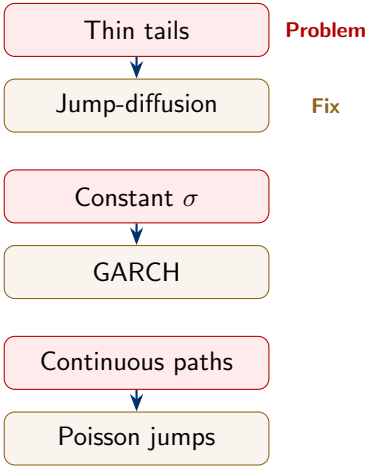
What's Wrong With Our Models?

The log-normal model assumes:

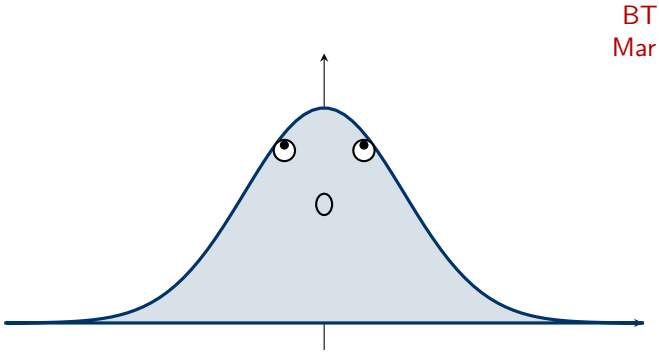
1. Returns are **normally distributed**
2. Volatility is **constant** over time
3. Price paths are **continuous** (no jumps)
4. Returns are **independent** across days

Reality of crypto returns:

1. Fat tails: kurtosis $\gg 3$
2. Volatility **clusters**: calm and stormy periods
3. Sudden **jumps**: flash crashes, liquidation cascades
4. r_t^2 is **autocorrelated**: today's big move predicts tomorrow's



The Normal Distribution's Nightmare



"That's not supposed to happen!"

Under a normal distribution, a -50% daily move is a $> 8\sigma$ event: probability $\approx 10^{-18}$

Today's Roadmap

Concepts (Slides 8–30)

- Log-normal model and its failures
- GARCH(1,1): estimation and forecasting
- Merton jump-diffusion: simulation
- Implied volatility and the smile

Hands-On (Slides 31–40)

- Fit GARCH to BTC returns in Python
- Forecast conditional volatility
- Simulation-based VaR
- Compare to historical VaR

Log-Normal Model: Quick Recap

Geometric Brownian Motion (physical measure)

$$\frac{dS_t}{S_t} = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$$

Solution:

$$S_T = S_0 \exp\Big[\big(\mu - \tfrac{\sigma^2}{2}\big)\,T + \sigma W_T\Big] \qquad \Longrightarrow \qquad \ln\Big(\frac{S_T}{S_0}\Big) \sim \mathcal{N}\Big(\big(\mu - \tfrac{\sigma^2}{2}\big)\,T,\ \sigma^2\,T\Big)$$

Key parameters:

- μ: drift (expected return per unit time)
- σ: volatility (constant!)
- The $-\sigma^2/2$ term is the Itô correction (convexity adjustment)

Estimation from data: given daily log-returns $r_t = \ln(S_t/S_{t-1})$,

$$\hat{\mu} = \frac{252}{n}\sum^{n} r_t, \qquad \hat{\sigma} = \sqrt{\frac{252}{n-1}\sum^{n}(r_t - \bar{r})^2}$$

What the Log-Normal Model Implies

Three core implications:

1. **Symmetric returns:** skewness $= 0$ (up and down equally likely)
2. **Thin tails:** kurtosis $= 3$ (excess kurtosis $= 0$)
3. **Constant volatility:** $\sigma_t = \sigma$ for all t

Additional implications:

- Returns on non-overlapping intervals are **independent**
- $|r_t|$ and r_t^2 have **no autocorrelation**
- **Continuous paths**: $\mathbb{P}(\text{jump} > x \text{ in } \Delta t) \to 0$ as $\Delta t \to 0$
- Volatility of 1-month returns $= \sigma\sqrt{1/12}$ (simple scaling)

Convenient but wrong

These properties make the model mathematically elegant (closed-form Black-Scholes, etc.) but they systematically fail for crypto returns.

Reality Check: BTC Returns Are Not Normal

BTC daily return statistics (2017–2025):

Statistic	BTC	Normal
Mean (daily)	0.06%	–
Std (daily)	3.8%	–
Skewness	−0.5	0
Kurtosis	12.3	3
Excess kurt.	9.3	0
Min return	−47%	–
Max return	+23%	–

What this means:

- **Negative skew**: left tail is heavier (crashes more extreme than rallies)
- **Kurtosis** $\gg 3$: tails are *much* fatter than normal
- A normal model with $\sigma = 3.8\%$ gives $\mathbb{P}(|r| > 15\%) \approx 10^{-5}$
- In practice, $|r| > 15\%$ occurs $\sim$5 times per year

Key Failure

The normal distribution underestimates tail risk by orders of magnitude.

Jarque-Bera Test for Normality

Jarque-Bera test statistic

$$JB = \frac{n}{6}\left(\hat{S}^2 + \frac{(\hat{K}-3)^2}{4}\right)$$

where $\hat{S}$ = sample skewness, $\hat{K}$ = sample kurtosis, n = sample size.

Under H_0: data is normally distributed, $JB \sim \chi^2(2)$.

Application to BTC daily returns (2017–2025, $n \approx 2{,}900$):

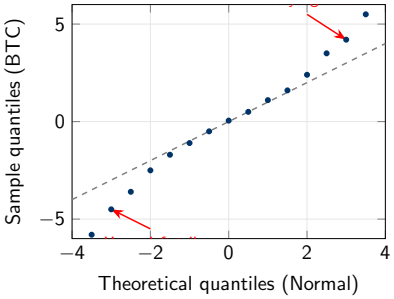
$$JB = \frac{2900}{6}\left((-0.5)^2 + \frac{(12.3-3)^2}{4}\right) = 483.3 \times (0.25 + 21.62) \approx 10{,}571$$

Critical value: $\chi^2_{0.001}(2) = 13.82$.

$$JB = 10{,}571 \gg 13.82 \qquad \Longrightarrow \qquad \textbf{Reject } H_0 \text{ at } p < 0.001$$

QQ-Plot: Seeing the Tail Deviation

Idea: plot sample quantiles vs. theoretical normal quantiles. If data is normal, points lie on the 45° line.



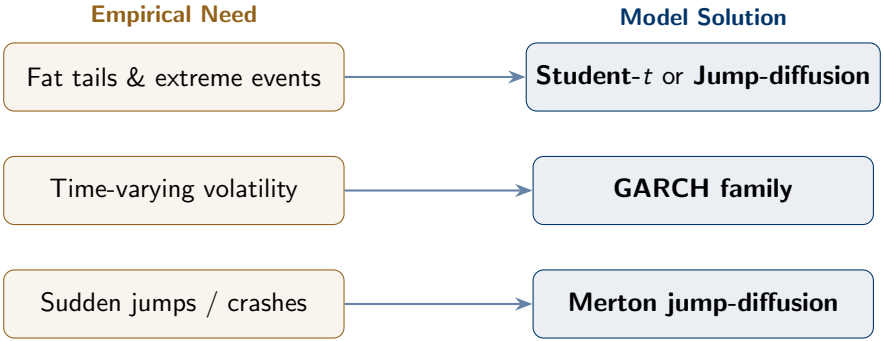
Reading a QQ-plot:

- Points on the line ⇒ data matches normal
- Points **below** line at left ⇒ left tail heavier than normal
- Points **above** line at right ⇒ right tail heavier than normal
- S-shape ⇒ fat tails on *both* sides

BTC pattern:

- Clear S-shape: both tails deviate
- Left tail deviates *more* (negative skew)
- Center is approximately normal

Takeaway: We Need Better Models

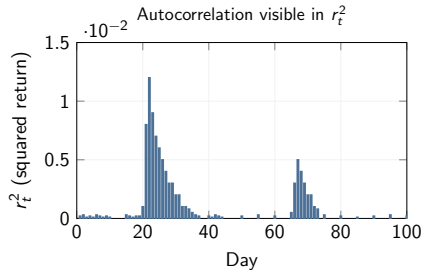


Rest of today:

1. GARCH: model time-varying volatility, estimate via MLE
2. Merton jump-diffusion: add sudden jumps, simulate paths
3. Implied volatility: read what the options market "thinks"

Volatility Clustering: The GARCH Motivation

Observation: large returns tend to follow large returns (of either sign).



Formalization:

- $\text{Corr}(r_t^2, r_{t-1}^2) > 0$ for crypto
- Autocorrelation of r_t^2 decays slowly
- Volatility is **persistent**: high vol today $\Rightarrow$ likely high vol tomorrow

Implication:

- Constant-σ models average over calm and stormy periods
- This *underestimates* risk in crises and *overestimates* in calm times
- We need σ_t that **adapts** to recent data

GARCH(1,1): The Workhorse Model [ ]

GARCH(1,1) specification

$$r_t = \mu + \varepsilon_t, \qquad \varepsilon_t = \sigma_t z_t, \quad z_t \sim \mathcal{N}(0,1) \tag{1}$$

$$\sigma_t^2 = \omega + \alpha\,\varepsilon_{t-1}^2 + \beta\,\sigma_{t-1}^2 \tag{2}$$

Parameters:

- $\omega > 0$: baseline variance (long-run anchor)
- $\alpha \geq 0$: weight on yesterday's squared shock (news coefficient)
- $\beta \geq 0$: weight on yesterday's conditional variance (persistence)
- Stationarity requires: $\alpha + \beta < 1$

Unconditional (long-run) variance:

$$\bar{\sigma}^2 = \frac{\omega}{1 - \alpha - \beta}$$

Typical BTC estimates: $\alpha \approx 0.10$, $\beta \approx 0.88$, so $\alpha + \beta = 0.98$ (very persistent).

GARCH Parameter Interpretation

α = **news reaction speed:**

- How fast σ_t responds to a shock
- High $\alpha \Rightarrow$ spiky vol
- Low $\alpha \Rightarrow$ smooth vol

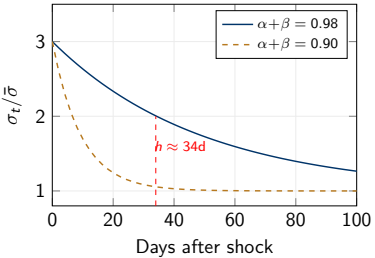
β = **persistence:**

- How slowly vol decays back to $\bar{\sigma}$
- High $\beta \Rightarrow$ long-lived vol clusters
- $\beta \approx 0.9$ is typical for financial data

Half-life of a volatility shock:

$$h = \frac{\ln 2}{\ln(\alpha+\beta)^{-1}} = \frac{\ln 2}{-\ln(\alpha+\beta)}$$

For $\alpha + \beta = 0.98$: $h = \frac{0.693}{0.02} \approx 34$ days



Crypto insight: BTC has $\alpha + \beta \approx 0.98$, so vol shocks take $\sim$5 weeks to halve. This explains why crisis periods feel long.

GARCH Estimation via Maximum Likelihood

Given: observed returns $r_1, \ldots, r_n$ and the model $r_t | \mathcal{F}_{t-1} \sim \mathcal{N}(\mu, \sigma_t^2)$.

Log-likelihood

$$\ell(\omega, \alpha, \beta) = -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{t=1}^{n} \left[\ln(\sigma_t^2) + \frac{(r_t - \mu)^2}{\sigma_t^2} \right]$$

where $\sigma_t^2 = \omega + \alpha(r_{t-1} - \mu)^2 + \beta \, \sigma_{t-1}^2$ is computed recursively.

Key insight: this is *not* a simple MLE — σ_t^2 depends on all past data through the recursion. No closed-form solution; must optimize numerically.

Python implementation:

```
from arch import arch_model
model = arch_model(returns, vol='GARCH', p=1, q=1)
result = model.fit(disp='off')
print(result.summary())
```

GARCH Volatility Forecasting

One-step-ahead forecast (known at time t):

$$\sigma_{t+1}^2 = \omega + \alpha\,\varepsilon_t^2 + \beta\,\sigma_t^2$$

Multi-step forecast (h days ahead):

$$\mathbb{E}_t[\sigma_{t+h}^2] = \bar{\sigma}^2 + (\alpha+\beta)^h\big(\sigma_t^2 - \bar{\sigma}^2\big)$$

where $\bar{\sigma}^2 = \frac{\omega}{1-\alpha-\beta}$ is the unconditional variance.

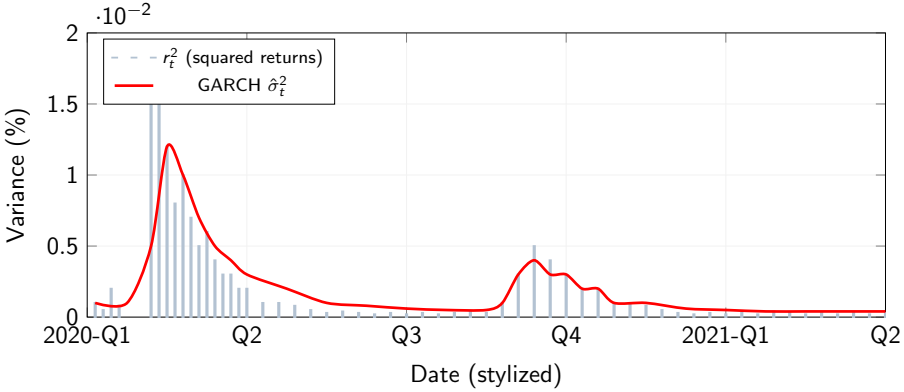
Interpretation:

- Forecast **mean-reverts** to $\bar{\sigma}^2$ at rate $(\alpha+\beta)^h$
- If current vol is high, forecast starts high and decays
- If current vol is low, forecast starts low and rises
- For large h: $\mathbb{E}_t[\sigma_{t+h}^2] \to \bar{\sigma}^2$ (converges to long-run)

30-day cumulative variance (for VaR)

$$\sum_{h=1}^{30} \mathbb{E}_t[\sigma_{t+h}^2] = 30\,\bar{\sigma}^2 + \frac{(\alpha+\beta)(1-(\alpha+\beta)^{30})}{1-(\alpha+\beta)} \cdot (\sigma_t^2 - \bar{\sigma}^2)$$

GARCH Conditional Variance vs. Squared Returns



Key observation: GARCH $\hat{\sigma}_t^2$ (red) tracks the **envelope** of squared returns, smoothing out the noise but capturing the volatility regime changes.

GARCH Limitations: What It Cannot Do

GARCH captures:

- ✓ Volatility clustering
- ✓ Heavy tails (via time-varying σ_t)
- ✓ Mean-reversion of volatility
- ✓ Conditional forecasts

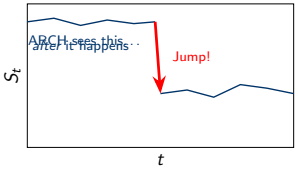
GARCH does NOT capture:

- × Sudden **jumps** (instantaneous large moves)
- × Asymmetric volatility response (leverage effect)[a]
- × Multi-day crash events (cascading liquidations)
- × Structural breaks

The fundamental issue:

GARCH is *backward-looking*: σ_{t+1} reacts to ε_t.

It cannot anticipate a jump that has not yet occurred.



$\Rightarrow$ We need **jump-diffusion** models.

Checkpoint

Quick Question

If $\alpha + \beta = 0.99$ instead of 0.98, does the half-life of a volatility shock approximately double?

Checkpoint

Quick Question

If $\alpha + \beta = 0.99$ instead of 0.98, does the half-life of a volatility shock approximately double?

Answer

Yes. Half-life $h = \ln 2/(-\ln(\alpha + \beta))$. For $\alpha + \beta = 0.98$: $h \approx \ln 2/0.0202 \approx 34$ days. For $\alpha + \beta = 0.99$: $h \approx \ln 2/0.01005 \approx 69$ days—roughly double. Higher persistence means shocks decay much more slowly.

Merton (1976): Smooth Path + Poisson Jumps

Core idea: combine continuous diffusion (GBM) with discrete jumps.

Components

1. **Diffusion**: smooth price changes (normal market fluctuations)
2. **Jumps**: sudden, discontinuous moves (crashes, news events, liquidation cascades)
3. Jumps arrive as a **Poisson process** (random, independent timing)

Poisson process $N(t)$ with intensity λ:

- $\mathbb{P}(1 \text{ jump in } \Delta t) = \lambda \Delta t$
- $\mathbb{P}(0 \text{ jumps in } \Delta t) = 1 - \lambda \Delta t$

What this adds to GBM:

- Path is *mostly* continuous
- Occasionally interrupted by sudden jumps
- Jump sizes are random (log-normal)
- Jumps generate the fat tails we observe

Crypto motivation

BTC experiences $\sim$4–8 large jumps per year ($> 5\%$ in a day). Set $\lambda \approx 6$: on average, one jump every 2 months.

Merton Jump-Diffusion: The Formula

Terminal price

$$S_T = S_0 \cdot \exp\Big[\big(\mu - \tfrac{\sigma^2}{2} - \lambda k\big)\,T + \sigma Z\sqrt{T}\Big] \cdot \prod_{i=1}^{N(T)} (1 + J_i)$$

where:

- $Z \sim \mathcal{N}(0,1)$: standard normal (diffusion component)
- $N(T) \sim \text{Poisson}(\lambda T)$: number of jumps in $[0, T]$
- J_i: random jump size, with $\ln(1 + J_i) \sim \mathcal{N}(\mu_J, \sigma_J^2)$
- $k = \mathbb{E}[J_i] = e^{\mu_J + \sigma_J^2/2} - 1$: expected jump size (compensator)
- The $-\lambda k$ term ensures $\mathbb{E}[S_T]$ is correct (drift compensation)

Five parameters: $\theta = (\mu, \sigma, \lambda, \mu_J, \sigma_J)$

Parameter	Meaning	Typical BTC value
σ	Diffusion volatility	50–60%

Jump Parameters: Building Intuition

λ = jump frequency:

- $\lambda = 0$: no jumps (plain GBM)
- $\lambda = 4$: expect 4 jumps/year
- $\lambda = 12$: about once per month

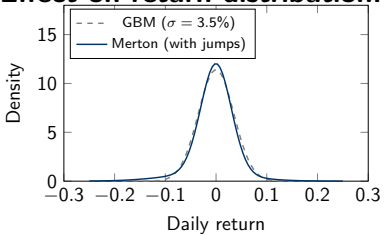
μ_J = average jump size:

- $\mu_J = 0$: jumps symmetric (up/down equal)
- $\mu_J < 0$: crash bias (more down-jumps)
- $\mu_J = -0.05$: average jump is $\approx -5\%$

σ_J = jump volatility:

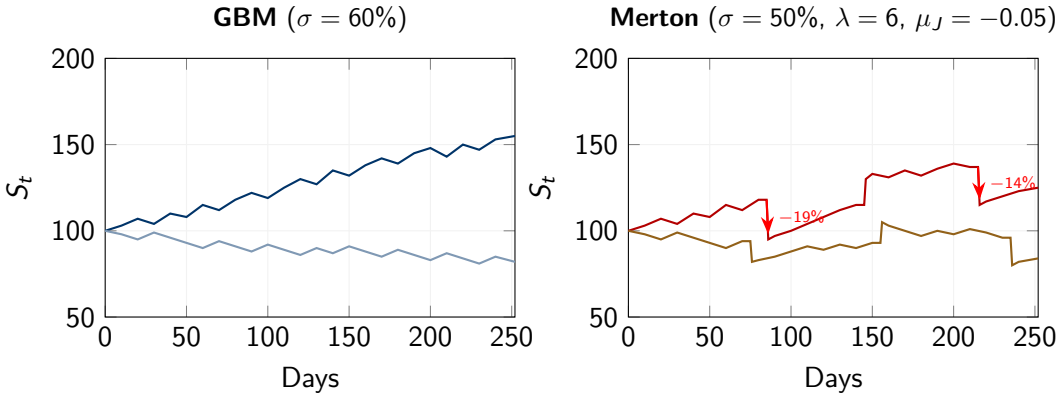
- Small σ_J: jumps have predictable size
- Large σ_J: jumps can be tiny or massive

Effect on return distribution:



Key insight: the Merton density is a **mixture** of normals with different variances, creating naturally heavier tails.

Simulation: GBM vs. Merton Jump-Diffusion



Key difference: GBM paths are smooth. Merton paths have sudden discontinuities (jumps) that match the crash-like behavior observed in crypto.

Monte Carlo Pricing with Jump-Diffusion

Goal: price a European option by simulating the risk-neutral distribution of S_T.

Algorithm

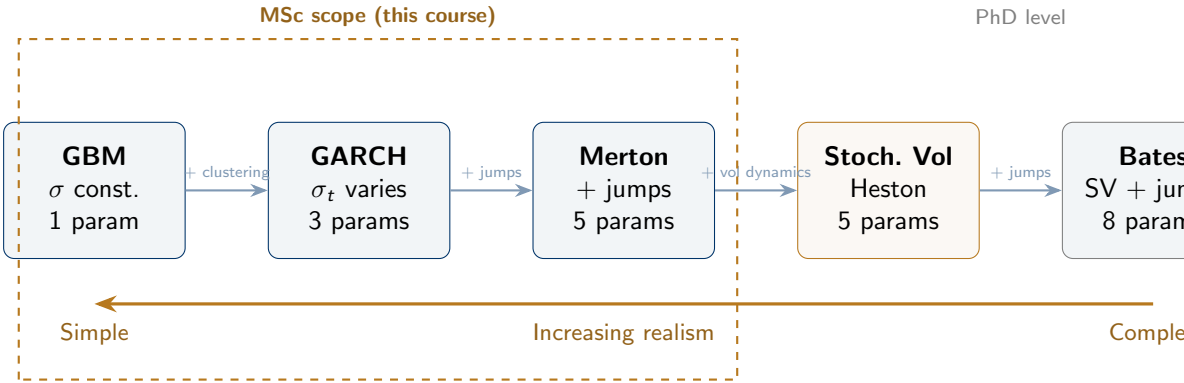
1. For $i = 1, \ldots, M$ (e.g., $M = 10{,}000$):
   1. Draw $Z_i \sim \mathcal{N}(0,1)$
   2. Draw $N_i \sim \text{Poisson}(\lambda T)$
   3. For each jump $j = 1, \ldots, N_i$: draw $J_{ij} \sim \mathcal{N}(\mu_J, \sigma_J^2)$
   4. Compute $S_T^{(i)} = S_0 \exp\big[(r - \sigma^2/2 - \lambda k)T + \sigma\sqrt{T}\, Z_i + \sum_j J_{ij}\big]$
2. Compute payoffs: $V_i = e^{-rT} \max(S_T^{(i)} - K, 0)$ (for a call)
3. Price: $\hat{C} = \frac{1}{M} \sum_{i=1}^{M} V_i$

Standard error: $\text{SE} = \frac{\hat{s}}{\sqrt{M}}$ where $\hat{s}$ is sample std of V_i.

Convergence: $M = 10{,}000$ gives SE $\approx 1\%$ of price. For 0.1% accuracy need $M = 10^6$.

Why Monte Carlo?

The Pricing Model Zoo



Tradeoff: more parameters $\Rightarrow$ better fit, but harder to estimate and more prone to overfitting. For MSc work, GARCH + Merton covers most practical needs.

Implied Volatility: Inverting Black-Scholes

Idea: use Black-Scholes *backwards*. Given a market price C_{mkt}, solve for σ_{impl}:

$$C_{\text{mkt}} = \text{BS}(S, K, T, r, \sigma_{\text{impl}})$$

Why?

- Apples-to-apples comparison across strikes/maturities
- Transforms dollar price into a volatility number
- If BS were correct, σ_{impl} would be the *same* for all K and T

Computation:

- No closed-form inversion
- Use Newton-Raphson:
  $\sigma^{(n+1)} = \sigma^{(n)} - \frac{\text{BS}(\sigma^{(n)}) - C_{\text{mkt}}}{\text{Vega}(\sigma^{(n)})}$

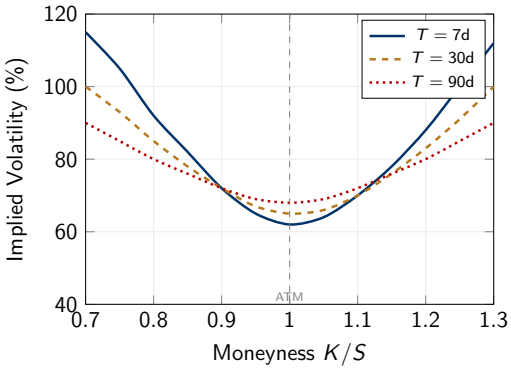
Python:

```
from scipy.optimize import brentq
def implied_vol(C_mkt, S, K, T, r):
  f = lambda s:  bs_call(S,K,T,r,s) - C_mkt
  return brentq(f, 0.01, 5.0)
```

Key insight

σ_{impl} is **not** a forecast of future vol. It is the market's risk-neutral expectation of vol, embedded in option prices.

The Volatility Smile: IV vs. Moneyness



Reading the smile:

- **U-shape**: OTM puts and OTM calls are more expensive than ATM
- **Short tenors**: steeper smile (more tail risk per day)
- **Long tenors**: flatter (jumps average out)

What causes the smile?

- Fat tails $\Rightarrow$ OTM options more valuable
- BS underprices them
- Market "corrects" by quoting higher IV at the wings

The Smile as a "Model Error Thermometer"

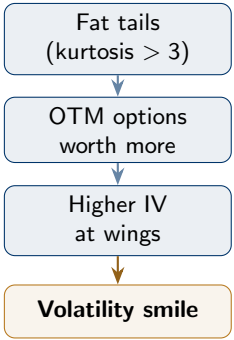
If Black-Scholes were correct:

- σ_{impl} would be flat across all strikes
- A single σ would price everything

The smile tells us:

- **Wings up** = "tails are fatter than normal"
- **Left wing higher** = "crash risk is underpriced by BS"
- **Steepness** $\propto$ departure from log-normality

Quantitative link:
Under a jump-diffusion model, the smile arises naturally because:



Practical use

Traders use the smile shape to choose which model to use. Steep smile $\Rightarrow$ need jumps.

Bitcoin DVOL: The Crypto "VIX"

What is DVOL?

- Deribit Volatility Index
- 30-day implied volatility for BTC options
- Methodology similar to CBOE VIX
- Computed from OTM puts and calls across strikes

Formula (simplified):

$$\text{DVOL}^2 = \frac{2}{T} \sum_i \frac{\Delta K_i}{K_i^2} e^{rT} Q(K_i)$$

where $Q(K_i)$ is the mid-price of OTM option at strike K_i.

Key levels:

DVOL vs. VIX:

	DVOL	VIX
Typical level	55%	17%
Crisis peak	180%	80%
Underlying	BTC	S&P 500
Market	Deribit	CBOE
Hours	24/7	Market hrs

Key observations:

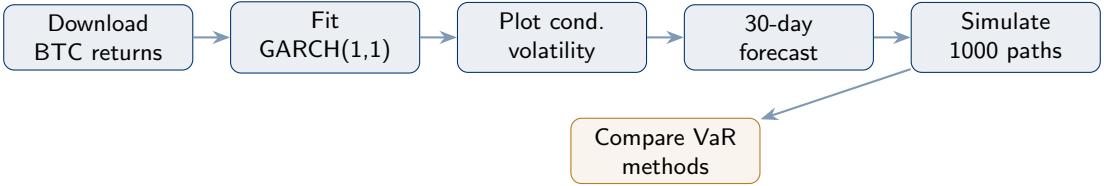
- BTC implied vol is $\sim 3\times$ equity vol
- DVOL mean-reverts (just like VIX)
- Negative correlation with BTC price (fear gauge)

Hands-On

GARCH Modeling for Crypto

Fit · Forecast · Simulate · Compare

Exercise Overview: From Data to Risk Forecast



Tools

- Python 3.10+
- `yfinance`: data download
- `arch`: GARCH estimation
- `numpy`, `matplotlib`: simulation & plots

Learning Objectives

- Estimate GARCH via MLE
- Interpret parameter estimates
- Forecast conditional volatility
- Compare VaR methodologies

Step 1: Download BTC Daily Returns (2020–2025)

Code

```
import yfinance as yf
import numpy as np

btc = yf.download('BTC-USD', start='2020-01-01', end='2025-12-31')
btc['log_ret'] = np.log(btc['Close'] / btc['Close'].shift(1))
returns = btc['log_ret'].dropna() * 100   # in percent
```

Quick diagnostics:

- Print `returns.describe()` — check mean, std, min, max
- Plot histogram: `returns.hist(bins=100)`
- Compute skewness and kurtosis: `returns.skew()`, `returns.kurtosis() + 3`
- Run Jarque-Bera: `scipy.stats.jarque_bera(returns)`

Expected output:

- ∼2,000 daily observations
- Std ≈ 3.5–4.0%
- Skewness ≈ −0.5, Kurtosis ≈ 10–14

Step 2: Fit GARCH(1,1) via MLE

Code

```
from arch import arch_model

model = arch_model(returns, mean='Constant', vol='GARCH', p=1, q=1)
result = model.fit(disp='off')
print(result.summary())
```

Reading the output:

Parameter	Estimate	Interpretation
ω	e.g., 0.15	Baseline variance level
α	e.g., 0.10	Yesterday's shock weight
β	e.g., 0.88	Persistence of variance
$\alpha + \beta$	e.g., 0.98	Total persistence

Check:

- Is $\alpha + \beta < 1$? (stationarity)

Step 3: Plot Conditional Volatility vs. Realized Vol

Code

```
cond_vol = result.conditional_volatility
realized_vol = returns.rolling(30).std() * np.sqrt(252)

import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(12, 4))
ax.plot(cond_vol.index, cond_vol * np.sqrt(252), label='GARCH')
ax.plot(realized_vol.index, realized_vol, label='30d Realized', alpha=0.7)
ax.legend(); ax.set_ylabel('Annualized Vol (%)')
plt.show()
```

What to look for:

- GARCH vol should **lead** realized vol (it reacts faster to shocks)
- Both should spike during known crises (March 2020, May 2021, Nov 2022)
- GARCH is smoother; realized vol is lagged (backward-looking window)
- Discrepancies highlight when GARCH under/over-estimates risk

Discussion: Why does GARCH react faster? (Answer: it uses yesterday's shock immediately,

Step 4: 30-Day Volatility Forecast

Code

```
forecasts = result.forecast(horizon=30)
var_forecast = forecasts.variance.iloc[-1].values   # 30-day path
vol_forecast = np.sqrt(var_forecast) * np.sqrt(252)   # annualized
```

Recall the formula:

$$\mathbb{E}_t[\sigma^2_{t+h}] = \bar{\sigma}^2 + (\alpha+\beta)^h\left(\sigma^2_t - \bar{\sigma}^2\right)$$

Plot: forecast vol over 30 days

- If current vol $>$ long-run: forecast **decreases**
- If current vol $<$ long-run: forecast **increases**
- All forecasts converge to $\bar{\sigma}$

Exercise questions:

- What is the long-run annualized vol?
- How many days until forecast is within 10% of $\bar{\sigma}$?
- How does the forecast change if you re-estimate using only 2024–2025 data?

Step 5: Simulate 1,000 GARCH Paths → VaR

Algorithm (pseudocode)

```
sigma2_t = result.conditional_volatility.iloc[-1]**2   # current var
for path in range(1000):
    cumulative_return = 0
    s2 = sigma2_t
    for day in range(30):
        z = np.random.standard_normal()
        r = mu + np.sqrt(s2) * z
        cumulative_return += r
        s2 = omega + alpha * (r - mu)**2 + beta * s2
    portfolio_returns[path] = cumulative_return
```

Value-at-Risk (VaR) from simulations:

$$\text{VaR}_{95\%}^{30d} = -\text{quantile}_{5\%}(\text{portfolio\_returns})$$

Interpretation: With 95% confidence, the portfolio will not lose more than VaR over 30 days (under the GARCH model).

Step 6: Compare GARCH VaR vs. Historical VaR

Three VaR methods:

Method	Formula	Assumes
Historical	5^{th} percentile of past 30d returns	Past = future
Parametric (Normal)	$\mu - 1.645 \cdot \sigma \cdot \sqrt{30}$	Constant σ, normality
GARCH simulation	5^{th} percentile of simulated paths	Time-varying σ

Expected results:

- In **calm periods**: GARCH VaR $<$ Historical VaR
  (GARCH knows current vol is low $\Rightarrow$ less risk)
- In **crisis periods**: GARCH VaR $>$ Historical VaR
  (GARCH reacts to recent spike; historical still includes old calm data)
- Parametric VaR is **always wrong** (underestimates in crises, overestimates in calm)

Discussion prompt

Which VaR method would you trust for a crypto portfolio? When would you prefer each?

Discussion: When to Use Which Model?

Does GARCH capture March 2020?

- GARCH **reacts** after the crash: σ_{t+1} spikes
- But it did **not predict** the -50% move
- A GARCH path rarely produces a -50% daily return
- Jump models handle this better

GARCH is best for:

- Short-term vol forecasting
- Risk management (VaR, ES)
- Understanding vol regimes
- As a building block in more complex models

Jump models are better for:

- Pricing OTM options
- Stress testing (extreme scenarios)
- Capturing tail risk
- When you need to explain the smile

In practice: combine both.

- GARCH-Jump models (GARCH with Poisson jumps)
- Use GARCH for "normal" vol dynamics
- Add jumps for tail events
- The `arch` package supports this via GJR-GARCH with Student-t innovations

Day 1: Summary and Key Takeaways

What we learned:

1. Log-normal model **fails** for crypto: fat tails, volatility clustering, jumps
2. **GARCH** captures time-varying volatility; estimated via MLE
3. **Merton jump-diffusion** adds sudden crashes; simulated via Monte Carlo
4. **Implied volatility** reveals what BS gets wrong (the smile)
5. DVOL: the crypto fear gauge

Key formulas:

- GARCH: $\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$
- Merton: $S_T = S_0 e^{(\mu-\sigma^2/2-\lambda k)T+\sigma Z\sqrt{T}} \cdot \prod(1+J_i)$
- Half-life: $h = \frac{\ln 2}{-\ln(\alpha+\beta)}$

Day 2 Preview

DeFi Protocol Design:

- Uniswap V3 concentrated liquidity
- Impermanent loss derivation
- Lending protocol economics

Reading: [1] (GARCH original), [2] Ch. 26 (volatility smiles).

References I

[1] Tim Bollerslev. "Generalized Autoregressive Conditional Heteroscedasticity". In: *Journal of Econometrics* 31.3 (1986), pp. 307–327.

[2] John C. Hull. *Options, Futures, and Other Derivatives*. 11th. Pearson, 2022.