

# Volatility Modeling: ARCH, GARCH and Extensions

ARCH effect | GARCH(1,1) | EGARCH | GJR-GARCH | Conditional volatility | VaR

Instructor

**Assoc. Prof. Dr. Onur POLAT**

# Week 4 — Agenda

*What we will cover today*

**01** Volatility Clustering and the ARCH Effect

**02** ARCH(q) Model — Engle (1982)

**03** GARCH(p,q) Model — Bollerslev (1986)

**04** GARCH Extensions: EGARCH, GJR-GARCH

**05** Model Estimation and Diagnostics

**06** Conditional Volatility Forecasting

**07** Value at Risk (VaR)

**08** Real-World Application: Full GARCH Pipeline

# Bridge from Week 3 to Week 4

*What ARIMA residuals reveal about the variance process*

## ARIMA Complete

We converted price to log-returns, fitted ARIMA(p,0,q) for the conditional mean equation.

## ARCH LM Test Result

Applied ARCH LM test to ARIMA residuals —  $p < 0.05$ : volatility clustering is present.

## Squared Residual ACF

The ACF of squared residuals shows significant lags — variance explains itself over time.

## Today's Goal

Model the conditional variance: forecast tomorrow's volatility with the GARCH family.

*ARIMA models the conditional mean but NOT the conditional variance. Financial series exhibit strong variance dependence. The GARCH family fills this gap: it simultaneously models both the mean and variance processes.*

## SECTION 1

# Volatility Clustering and the ARCH Effect

*Mandelbrot (1963) — Engle (1982) — Stylized facts of financial returns*

# Volatility Clustering

*Mandelbrot (1963): "Large changes tend to be followed by large changes"*

- Observation: In financial markets, periods of high volatility cluster together — calm periods follow calm, turbulent periods follow turbulent.
- The variance of returns changes over time (heteroskedastic), violating the OLS assumption of constant error variance.
- Implications: OLS t-statistics and p-values become unreliable; standard confidence intervals are too narrow in volatile regimes.
- When we model conditional variance, we obtain both better forecasts and more reliable risk measures.
- ARCH/GARCH motivation: VaR and CVaR calculations require conditional volatility estimates. Basel II/III capital adequacy relies on this.



# Stylized Facts of Financial Returns

*Empirical regularities observed across markets, assets, and time periods*

## Universal Patterns in Financial Data

### ***Fat Tails (Leptokurtosis)***

Return distributions have heavier tails than the Normal — extreme events occur more frequently than predicted by Gaussian models.

### ***Volatility Clustering***

Periods of high (low) volatility tend to persist — large absolute returns cluster in time. Autocorrelation in squared returns.

### ***Leverage Effect***

Negative returns increase volatility more than positive returns of the same magnitude (Black, 1976). Bad news hits harder.

### ***Mean Reversion***

Volatility eventually reverts to a long-run average level, but the speed varies — crisis periods revert slowly.

### ***Asymmetry***

The impact of a shock on volatility depends on its sign: stock price drops raise leverage ratios, increasing riskiness.

# Real-World Volatility Events

*Why conditional variance matters: evidence from global markets*

**80%+**

VIX during  
2008 GFC

**82.7**

VIX intraday peak  
March 16, 2020

**20%**

BIST-100 avg.  
annual volatility

**50%+**

BTC realized vol  
typical year

| Event                   | Period              | Daily Vol (ann.) | Key Feature               |
|-------------------------|---------------------|------------------|---------------------------|
| Global Financial Crisis | Sep 2008 - Mar 2009 | ~80%             | Persistent high vol       |
| COVID-19 Crash          | Feb - Apr 2020      | ~85%             | Sudden spike, fast revert |
| Turkish Lira Crisis     | Aug 2018            | ~60%             | EM contagion risk         |
| Oil Price War           | Mar 2020            | ~150%            | WTI went negative         |
| Bitcoin Crash           | May 2021            | ~120%            | Crypto leverage unwind    |

## SECTION 2

# ARCH(q) Model

*Autoregressive Conditional Heteroskedasticity — Engle (1982)*

**Nobel Prize in Economics 2003**

# ARCH(q) Model — Engle (1982)

*Nobel Prize 2003: "Modeling time-varying volatility"*

$$\text{Mean: } r_t = \mu + \varepsilon_t \quad \text{Variance: } \sigma^2_t = \omega + \alpha_1 \varepsilon^2_{t-1} + \alpha_2 \varepsilon^2_{t-2} + \dots + \alpha_q \varepsilon^2_{t-q}$$

*The conditional variance depends on past squared shocks*

## ARCH(q) Parameters and Constraints

$$\omega > 0$$

Intercept — ensures baseline positive variance; represents long-run variance floor.

$$\alpha_i \geq 0$$

Impact of past shocks — each coefficient must be non-negative to keep variance positive.

$$\sum \alpha_i < 1$$

Stationarity condition — ensures variance does not explode to infinity. Guarantees mean reversion.

$$\varepsilon_t = \sigma_t \cdot z_t$$

Error decomposition:  $z_t \sim N(0,1)$  or  $t(v)$ . Separates volatility from standardized innovation.

### **Unconditional var**

$\sigma^2 = \omega / (1 - \sum \alpha_i)$ . The long-run average level to which conditional variance reverts.

# ARCH LM Test — Pre-diagnostic Step

*Mandatory check before fitting any GARCH model*

- Step 1: Estimate ARIMA(p,0,q) on log-returns (Week 3)
- Step 2: Extract residuals:  $\varepsilon_t = r_t - \hat{r}_t$
- Step 3: Compute squared residuals:  $\varepsilon_t^2$
- Step 4: Plot ACF of  $\varepsilon_t^2$  — check for autocorrelation
- Step 5: Apply ARCH LM test — formal decision
- $H_0$ : No ARCH effect (residuals are homoskedastic)
- Test statistic:  $T \cdot R^2 \sim \chi^2(q)$
- $p < 0.05 \rightarrow$  Reject  $H_0 \rightarrow$  ARCH/GARCH model needed

## Python

```
from statsmodels.stats.diagnostic import
het_arch
from statsmodels.tsa.arima.model import
ARIMA

# Fit ARIMA → extract residuals
arima_fit = ARIMA(log_ret,
order=(1,0,1)).fit()
resid = arima_fit.resid

# ARCH LM test — lag=10
arch_lm = het_arch(resid, nlags=10)
print(f"ARCH LM statistic:
{arch_lm[0]:.4f}")
print(f"ARCH LM p-value :
{arch_lm[1]:.4f}")
if arch_lm[1] < 0.05:
    print("ARCH effect present -> fit
GARCH")
```

# ARCH( $q$ ) Limitations and Motivation for GARCH

## *Why we need a more parsimonious model*

- Problem 1: High order  $q$  required — to capture long memory in volatility, ARCH needs many lags ( $q = 20+$ ), meaning many parameters to estimate.
- Problem 2: Parsimony — with limited sample sizes, estimating 20+ parameters is unreliable and may lead to overfitting.
- Problem 3: Non-negativity — all  $\alpha_i$  must be non-negative, which becomes hard to enforce with many parameters.
- Problem 4: No asymmetry — ARCH treats positive and negative shocks identically, but markets react differently to good vs. bad news.
- Solution: GARCH(1,1) captures the same long-memory structure with just 3 parameters ( $\omega, \alpha, \beta$ ), acting like an infinite-order ARCH.
- Analogy: ARMA vs. AR — just as ARMA extends AR with MA terms for parsimony, GARCH extends ARCH with lagged variance terms.

## SECTION 3

# GARCH(p,q) Model

*Generalized ARCH — Bollerslev (1986)*

# GARCH(1,1) — The Gold Standard

*The most widely used volatility model in finance*

$$\sigma^2_t = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta \cdot \sigma^2_{t-1}$$

*Variance = Intercept + ARCH term (short-run shock) + GARCH term (persistence)*

## $\omega$ (omega)

Long-run variance weight.  
 $\omega > 0$  required.  
Unconditional variance:  
 $\bar{\sigma}^2 = \omega / (1 - \alpha - \beta)$

## $\alpha \cdot \varepsilon^2_{t-1}$

ARCH term (short-run shock).  
 $\alpha \geq 0$  required.  
Large  $\alpha$ : fast reaction to shocks.  
Reflects news impact.

## $\beta \cdot \sigma^2_{t-1}$

GARCH term (persistence).  
 $\beta \geq 0$  required.  
Large  $\beta$ : volatility stays high.  
Captures memory in variance.

*Constraints:  $\omega > 0$ ,  $\alpha \geq 0$ ,  $\beta \geq 0$ ,  $\alpha + \beta < 1$  (stationarity) | Interpretation:  $\alpha + \beta$  close to 1  $\rightarrow$  high volatility persistence |  $\alpha + \beta = 1 \rightarrow$  IGARCH (permanent shocks)*

# GARCH(1,1) — Python with the arch Library

```
from arch import arch_model
# arch library GARCH(1,1) with Normal errors
model_n = arch_model(log_ret * 100, # percent scale
    vol='GARCH', p=1, q=1,
    dist='normal', mean='Constant')
result_n = model_n.fit(dispatch='off')
print(result_n.summary())
# GARCH(1,1) with Student-t errors (fat tails)
model_t = arch_model(log_ret * 100,
    vol='GARCH', p=1, q=1,
    dist='t', mean='Constant')
result_t = model_t.fit(dispatch='off')
# Compare via AIC
print(f"Normal AIC: {result_n.aic:.2f}")
print(f"t-dist AIC: {result_t.aic:.2f}")
winner = "t-dist" if result_t.aic < result_n.aic else
"Normal"
print(f"Winner: {winner}")
# Interpret parameters
p = result_t.params
alpha, beta = p['alpha[1]'], p['beta[1]']
print(f"alpha+beta = {alpha+beta:.4f} (persistence)")
unc_var = p['omega'] / (1 - alpha - beta)
print(f"Unconditional vol (ann.):
{ (unc_var**0.5*252**0.5/100) :.2%}")
```

## Key Points

- pip install arch (Kevin Sheppard)
- Scale returns to % for numerical stability
- vol='GARCH', p=1 (GARCH lag), q=1 (ARCH lag)
- dist options: normal, t, ged, skewt
- Student-t almost always wins for financial data
- $\alpha + \beta < 1$  ensures stationarity
- High persistence ( $> 0.95$ ) = slow mean reversion
- Unconditional vol = long-run average level

# Interpreting GARCH(1,1) Output

*What each parameter tells us about the volatility process*

| Parameter             | Typical Value              | Interpretation       | Implication                                     |
|-----------------------|----------------------------|----------------------|-------------------------------------------------|
| $\omega$ (omega)      | 0.01 – 0.10                | Baseline variance    | Higher $\rightarrow$ higher floor volatility    |
| $\alpha$ (alpha)      | 0.05 – 0.15                | Shock reactivity     | Higher $\rightarrow$ sharper spike on news      |
| $\beta$ (beta)        | 0.80 – 0.95                | Variance persistence | Higher $\rightarrow$ volatility decays slower   |
| $\alpha + \beta$      | 0.90 – 0.99                | Total persistence    | $> 0.99$ suggests IGARCH                        |
| $\nu$ (nu for t-dist) | 4 – 10                     | Tail heaviness       | Lower $\rightarrow$ fatter tails, more extremes |
| Half-life             | $\ln(2)/\ln(\alpha+\beta)$ | Days to decay 50%    | 10–70 days typical for equities                 |

Half-life formula:  $HL = \ln(2) / \ln(\alpha+\beta)$ . Example: if  $\alpha+\beta = 0.97$ ,  $HL = \ln(2)/\ln(0.97) \approx 23$  days. After a shock, it takes  $\sim 23$  trading days for the conditional variance to decay halfway to its long-run level.

## SECTION 4

# GARCH Extensions: EGARCH and GJR-GARCH

*Asymmetric volatility — modeling the leverage effect*

# The Leverage Effect

*Black (1976): Why negative shocks increase volatility more*

- Observation: A stock price drop increases the firm's debt-to-equity ratio (leverage), making the firm riskier and increasing future volatility.
- Consequence: Negative returns increase volatility more than positive returns of the same magnitude — this is asymmetric volatility.
- Evidence: For the S&P 500, a -2% daily return increases next-day volatility roughly 50% more than a +2% return.
- Standard GARCH(1,1) cannot capture this: it squares the error term, losing the sign information.
- EGARCH and GJR-GARCH add a parameter ( $\gamma$ ) to capture this asymmetry — one of the most important improvements.

Positive Shock (+2%) → Volatility rises by  $\alpha$

Negative Shock (-2%) → Volatility rises by  $\alpha + \gamma$

# EGARCH Model — Nelson (1991)

*Exponential GARCH: asymmetry without non-negativity constraints*

$$\ln(\sigma^2_t) = \omega + \alpha \cdot |z_{t-1}| + \gamma \cdot z_{t-1} + \beta \cdot \ln(\sigma^2_{t-1})$$

$z_{t-1} = \varepsilon_{t-1} / \sigma_{t-1}$  (standardized residual)

## $\alpha$ (ARCH)

Magnitude effect —  $|z_{t-1}|$  gives symmetric contribution. Both positive and negative shocks increase volatility.

## $\gamma$ (Asymmetry)

Leverage effect:  $\gamma < 0$  means negative shocks increase volatility more. Captures Black (1976) asymmetry directly.

## $\beta$ (Persistence)

AR coefficient on  $\ln(\sigma^2)$ . Stationarity requires  $|\beta| < 1$  only — no sign constraints on  $\omega$  or  $\alpha$ .

*EGARCH Advantages: (1) Log form  $\rightarrow$  no non-negativity constraints (2) Asymmetry modeled directly via  $\gamma$  (3) Consistent with Black's leverage observation*

# GJR-GARCH — Glosten, Jagannathan, Runkle (1993)

*Threshold GARCH: extra weight on negative shocks via indicator function*

$$\sigma^2_t = \omega + \alpha \cdot \varepsilon^2_{t-1} + \gamma \cdot \varepsilon^2_{t-1} \cdot I_{t-1} + \beta \cdot \sigma^2_{t-1}$$

$I_{t-1} = 1$  if  $\varepsilon_{t-1} < 0$  (bad news),  $I_{t-1} = 0$  if  $\varepsilon_{t-1} \geq 0$  (good news)

**Positive Shock ( $\varepsilon_{t-1} > 0$ ) →  $I = 0$**

$$\sigma^2_t = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta \cdot \sigma^2_{t-1}$$

Standard GARCH(1,1) effect — volatility increases by  $\alpha$  only.

**Negative Shock ( $\varepsilon_{t-1} < 0$ ) →  $I = 1$**

$$\sigma^2_t = \omega + (\alpha + \gamma) \cdot \varepsilon^2_{t-1} + \beta \cdot \sigma^2_{t-1}$$

Extra volatility from  $\gamma > 0$  — bad news has amplified impact.

*Leverage Effect: Black (1976) observed that stock price drops increase the firm's leverage ratio, raising riskiness → volatility rises more. GJR-GARCH captures this via  $\gamma > 0$ . Stationarity:  $\alpha + \gamma/2 + \beta < 1$ .*

# GARCH Family — Model Comparison

*Which model to use when?*

| Model      | Variance Equation                                                             | Asymmetry | Leverage | Use Case             |
|------------|-------------------------------------------------------------------------------|-----------|----------|----------------------|
| ARCH(q)    | $\omega + \sum \alpha_i \varepsilon_{t-i}^2$                                  | X         | X        | Initial analysis     |
| GARCH(1,1) | $\omega + \alpha \varepsilon_t^2 + \beta \sigma^2$                            | X         | X        | Standard forecasting |
| EGARCH     | $\ln(\sigma^2) = \omega + \alpha  z  + \gamma z + \beta \ln(\sigma^2)$        | ✓         | ✓        | Equities, FX         |
| GJR-GARCH  | $\omega + \alpha \varepsilon_t^2 + \gamma \varepsilon_t^2 I + \beta \sigma^2$ | ✓         | ✓        | Indices, crises      |
| IGARCH     | $\alpha + \beta = 1$ constrained                                              | X         | X        | Long-memory vol      |
| GARCH-M    | $r_t = \mu + \lambda \sigma_t + \varepsilon_t$                                | X         | X        | Risk-return tradeoff |

*Recommendation: Start with GARCH(1,1)-t as baseline, then test EGARCH-t and GJR-GARCH-t. Compare using AIC/BIC. For equity markets, asymmetric models almost always dominate.*

# EGARCH and GJR-GARCH — Python Estimation

## Asymmetric Models

```
from arch import arch_model
ret100 = log_ret * 100

# — EGARCH(1,1) with Student-t —
egarch = arch_model(ret100,
                    vol='EGARCH', p=1, q=1,
                    dist='t', mean='Constant')
eg_fit = egarch.fit(disp='off')
print("EGARCH params:", eg_fit.params.round(4))

# — GJR-GARCH(1,1) with Student-t —
gjrgarch = arch_model(ret100,
                    vol='GJR-GARCH', p=1, q=1, # o=1 for asymmetry
                    dist='t', mean='Constant')
gjrg_fit = gjrgarch.fit(disp='off')
print("GJR params:", gjrg_fit.params.round(4))
```

## Model Selection

```
# — Model Selection: AIC Comparison —
models = {
    'GARCH-Normal': arch_model(ret100,
                              vol='GARCH', p=1, q=1,
                              dist='normal'),
    'GARCH-t': arch_model(ret100,
                          vol='GARCH', p=1, q=1,
                          dist='t'),
    'EGARCH-t': arch_model(ret100,
                           vol='EGARCH', p=1, q=1,
                           dist='t'),
    'GJR-t': arch_model(ret100,
                        vol='GJR-GARCH', p=1, q=1,
                        dist='t'),
}
results = {k: v.fit(disp='off')
           for k, v in models.items()}
```

**Error Distributions:** 'normal' — standard, no fat tails | 't' — Student-t, fat tails via  $\nu$  | 'ged' — Generalized Error Distribution | 'skewt' — Skewed t, tails + asymmetry. For financial data, 't' or 'skewt' almost always fits better than 'normal'.

## SECTION 5

# Model Estimation and Diagnostics

*Maximum Likelihood | Residual checks | Model selection criteria*

# Maximum Likelihood Estimation (MLE)

*How GARCH parameters are estimated*

- GARCH models are estimated via Maximum Likelihood Estimation (MLE), not OLS.
- The log-likelihood function depends on the assumed error distribution (Normal, Student-t, GED).
- For Normal:  $L = -0.5 \cdot \sum [\ln(2\pi) + \ln(\sigma^2_t) + \varepsilon^2_t / \sigma^2_t]$
- The optimizer iterates until convergence — arch uses SLSQP or L-BFGS-B algorithms.
- Initial parameter values matter: poor starting points can lead to local optima or convergence failure.
- Model selection: AIC =  $-2L + 2k$ , BIC =  $-2L + k \cdot \ln(n)$ . Lower is better. BIC penalizes complexity more.

**AIC**

Akaike Information

**BIC**

Bayesian Information

**LogLik**

Log-Likelihood

# GARCH Residual Diagnostics

*Standardized residuals must behave like white noise*

## Standardized Residual ACF

**H<sub>0</sub>: No autocorrelation in  $\hat{z}_t$**

Plot ACF of  $\hat{z}_t = \varepsilon_t / \sigma_t$ . Should stay within confidence bands.

**If failed:**

If ACF significant: mean equation misspecified.

## Ljung-Box on $\hat{z}_t^2$

**H<sub>0</sub>: No autocorrelation in  $\hat{z}_t^2$**

Tests whether ARCH effect is fully captured.  $p > 0.05$  means the variance model is adequate.

**If failed:**

If  $p < 0.05$ : try higher  $p, q$  or asymmetric model.

## QQ-Plot / Distribution Fit

**H<sub>0</sub>:  $\hat{z}_t$  follows chosen distribution**

Plot QQ against the specified distribution (Normal, t). Points should align with the 45-degree line.

**If failed:**

Fat tails with Normal: switch to Student-t.

# Diagnostic Checks — Python Code

```
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.graphics.tsaplots import plot_acf
from scipy import stats
import matplotlib.pyplot as plt

# Standardized residuals
std_resid = gjr_fit.std_resid

# 1. ACF of standardized residuals
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
plot_acf(std_resid, lags=30, ax=ax1, title="Std. Resid. ACF")
plot_acf(std_resid**2, lags=30, ax=ax2, title="Std. Resid2 ACF (ARCH check)")
plt.tight_layout(); plt.show()

# 2. Ljung-Box on squared standardized residuals
lb_sq = acorr_ljungbox(std_resid**2, lags=[10, 20], return_df=True)
print(lb_sq[['lb_stat', 'lb_pvalue']])
# p > 0.05 -> ARCH effect cleaned

# 3. QQ-plot against Student-t
nu = gjr_fit.params['nu']
fig, ax = plt.subplots(figsize=(6, 6))
stats.probplot(std_resid, dist='t', sparams=(nu,), plot=ax)
ax.set_title(f'QQ Plot - Student-t (nu={nu:.1f})')
plt.tight_layout(); plt.show()

# 4. Jarque-Bera on standardized residuals
jb_stat, jb_p = stats.jarque_bera(std_resid)
print(f"Jarque-Bera stat: {jb_stat:.2f}, p={jb_p:.4f}")
```

## SECTION 6

# Conditional Volatility Forecasting

*In-sample fitted values | Multi-step ahead forecasts | Regime analysis*

# Volatility Forecasting — Theory

*How past shocks propagate into future variance estimates*

- 1-step forecast:  $\sigma^2_{t+1} = \omega + \alpha \cdot \varepsilon^2_t + \beta \cdot \sigma^2_t$  — uses the most recent shock and variance.
- h-step forecast: Converges to the unconditional variance  $\bar{\sigma}^2 = \omega / (1 - \alpha - \beta)$  as h increases.
- If  $\alpha + \beta$  is high (e.g. 0.98): convergence is slow — volatility stays elevated for weeks after a shock.
- If  $\alpha + \beta$  is low (e.g. 0.85): convergence is fast — the effect of a shock dissipates quickly.
- Annualization:  $\sigma_{\text{annual}} = \sigma_{\text{daily}} \times \sqrt{252}$ . Converts daily conditional volatility to the more interpretable annual scale.
- Forecasting ability: GARCH forecasts the level of volatility, not its direction. It's about how much, not which way.



# Volatility Forecasting — Python

## Forecasting Pipeline

```
# Conditional volatility (in-sample fitted values)
cond_vol = gjr_fit.conditional_volatility # daily, in %
cond_vol_annual = cond_vol * np.sqrt(252) / 100
# Plot returns vs conditional volatility
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8), sharex=True)
log_ret.plot(ax=ax1, color='#1A2A5E', lw=0.8, label='Log>Returns')
ax1.set_title('Log>Returns (%)')
ax1.legend()
cond_vol_annual.plot(ax=ax2, color='#C9922A', lw=1.2,
                    label='Conditional Volatility (Annualized)')
ax2.axhline(cond_vol_annual.mean(), color='#0D7A8C',
            ls='--', alpha=0.7, label=f'Mean: {cond_vol_annual.mean():.1%}')
ax2.set_title('GJR-GARCH(1,1) Conditional Volatility')
ax2.legend()
plt.tight_layout(); plt.show()
# 30-day ahead forecast
forecasts = gjr_fit.forecast(horizon=30, reindex=False)
var_fore = forecasts.variance.iloc[-1]
vol_fore = np.sqrt(var_fore) * np.sqrt(252) / 100
print(f"Average 30-day forecast vol: {vol_fore.mean():.1%}")
```

# Volatility Regime Analysis

## Identifying low / medium / high volatility regimes

- Low regime:  $\sigma_t < 0.75 \times \bar{\sigma}$  — calm market, low risk environment
- Medium regime:  $0.75\bar{\sigma} \leq \sigma_t \leq 1.5\bar{\sigma}$  — normal conditions
- High regime:  $\sigma_t > 1.5 \times \bar{\sigma}$  — crisis / stress period
- Practical use: Adjust portfolio risk dynamically based on regime
- More sophisticated: Markov-Switching GARCH (Hamilton, 1989) — probabilistic regime transitions

### Regime Code

```
# Rule-based regime classification
sigma_ann = gjr_fit.conditional_volatility
\
    * np.sqrt(252) / 100
sigma_mean = sigma_ann.mean()

def classify_regime(s, mean):
    if s < mean * 0.75: return 'Low'
    elif s < mean * 1.5: return 'Medium'
    else: return 'High'
```

```
regimes = sigma_ann.apply(
```

| Event               | Period       | Regime | Ann. Vol |
|---------------------|--------------|--------|----------|
| COVID-19 Crash      | Feb–Apr 2020 | HIGH   | 80–85%   |
| Turkish Lira Crisis | Aug 2018     | HIGH   | 55–60%   |
| Calm Bull Market    | 2017–2018 H1 | LOW    | 10–15%   |
| Normal Trading      | Most of 2019 | MEDIUM | 20–25%   |

## SECTION 7

# Value at Risk (VaR)

*GARCH-based dynamic risk measurement | Basel II/III internal models*

# Value at Risk — Definition and Methods

*Maximum loss estimate at a given confidence level over a time horizon*

$$P(r_t \leq -\text{VaR}_\alpha) = \alpha \rightarrow \text{VaR}_\alpha = -(\mu + z_\alpha \cdot \sigma_t)$$

*1-day 99% VaR: probability of exceeding this loss is 1% (1 day out of 100)*

| Method                              | Formula                                          | Distribution   | Pros                            | Cons                      |
|-------------------------------------|--------------------------------------------------|----------------|---------------------------------|---------------------------|
| Parametric<br>(Variance-Covariance) | $\text{VaR} = \mu + z_\alpha \cdot \sigma$       | Normal assumed | Simple, fast                    | Ignores fat tails         |
| GARCH-based<br>(Dynamic)            | $\text{VaR}_t = \mu_t + z_\alpha \cdot \sigma_t$ | Normal, t, GED | Adapts to market                | Model dependent           |
| Historical<br>Simulation            | $q_\alpha(\text{past returns})$                  | Non-parametric | No distribution assumption      | Slow to adapt             |
| Monte Carlo<br>Simulation           | $q_\alpha(\text{simulated paths})$               | Any            | Flexible, handles non-linearity | Computationally expensive |

# Why GARCH-based VaR?

*Dynamic risk measurement that adapts to changing market conditions*

## Static VaR Problem

Uses constant  $\sigma$  from historical window.

Calm period: VaR too high  $\rightarrow$  excess capital held.

Crisis period: VaR too low  $\rightarrow$  insufficient capital.

Risk measure lags behind reality.

## GARCH-VaR Solution

Uses time-varying  $\sigma_t$  from GARCH.

Volatility rises  $\rightarrow$  VaR automatically widens.

Volatility falls  $\rightarrow$  VaR contracts.

Risk measure is forward-looking.

- Basel II/III: Internal Models Approach (IMA) accepts GARCH-based VaR for regulatory capital calculations.
- Backtesting: Kupiec LR test validates VaR — compares actual vs. predicted exceedances over a rolling window.
- Expected Shortfall (ES / CVaR): Average loss beyond VaR threshold — provides the mean tail risk, not just the cutoff.
- Use Student-t quantiles instead of Normal when the model uses t-distributed errors — this better captures tail risk.

# GARCH-VaR — Python Implementation

## VaR + Backtesting

```
from scipy import stats

# Conditional volatility from fitted model
sigma_t = gjr_fit.conditional_volatility / 100
mu_t     = gjr_fit.params['Const'] / 100
nu       = gjr_fit.params['nu']    # t-dist df

# 99% VaR (1-day) using Student-t quantile
alpha = 0.01
t_q = stats.t.ppf(alpha, df=nu)
VaR_99 = -(mu_t + t_q * sigma_t)

# 95% VaR
VaR_95 = -(mu_t + stats.t.ppf(0.05, df=nu) * sigma_t)

# For 1M TL portfolio
portfolio = 1_000_000
VaR_TL = VaR_99.iloc[-1] * portfolio
print(f"Latest 99% VaR: {VaR_99.iloc[-1]*100:.2f}%")
print(f"    = {VaR_TL:,.0f} TL on 1M portfolio")

# Backtesting — Kupiec unconditional coverage
breaches = (log_ret < -VaR_99).sum()
```

## SECTION 8

# Real-World Application: Full GARCH Pipeline

*BIST-100 end-to-end | S&P 500 | Bitcoin | Multi-asset*

# BIST-100 Pipeline — Steps 1–2: Data and ARCH Test

*Lab exercise: end-to-end volatility modeling*

## Step 1: Data

```
# Step 1: Data Download and Preprocessing
import yfinance as yf
import numpy as np, pandas as pd
from arch import arch_model
from statsmodels.stats.diagnostic import
het_arch

bist = yf.download("XU100.IS",
                  start="2018-01-01", end="2024-12-31",
                  auto_adjust=True)['Close'].dropna()
log_ret = np.log(bist / bist.shift(1)).dropna()
* 100
print(f"Series length: {len(log_ret)} days")
print(f"Mean: {log_ret.mean():.4f}%")
print(f"Std: {log_ret.std():.4f}%")
print(f"Skew: {log_ret.skew():.4f}")
print(f"Kurt: {log_ret.kurtosis():.4f}")
```

## Step 2: ARCH Test

```
# Step 2: ARCH LM Test
from statsmodels.tsa.arima.model import ARIMA

# Mean equation via ARIMA(1,0,1)
arima_fit = ARIMA(log_ret, order=(1,0,1)).fit()
resid = arima_fit.resid

# ARCH LM test — lags 5, 10, 20
for lag in [5, 10, 20]:
    stat, p, _, _ = het_arch(resid, nlags=lag)
    status = "ARCH" if p < 0.05 else "No ARCH"
    print(f"Lag {lag:2d}: stat={stat:.2f}"
          f" p={p:.4f} -> {status}")

# Plot squared residual ACF
from statsmodels.graphics.tsaplots \
    import plot_acf
plot_acf(resid**2, lags=30,
         title="Squared Residuals ACF")
```

# BIST-100 Pipeline — Steps 3–4: Grid Search and Estimation

*Compare GARCH family models and select the best*

## Step 3: Grid Search

```
# Step 3: Model Grid Search
import warnings;
warnings.filterwarnings('ignore')
models = {
    'GARCH-N': arch_model(log_ret, vol='GARCH',
                          p=1, q=1, dist='normal'),
    'GARCH-t': arch_model(log_ret, vol='GARCH',
                          p=1, q=1, dist='t'),
    'EGARCH-t': arch_model(log_ret,
                           vol='EGARCH',
                           p=1, q=1, dist='t'),
    'GJR-t': arch_model(log_ret, vol='GARCH',
                        p=1, o=1, q=1, dist='t'),
}
fits = {k: v.fit(disps='off')
        for k, v in models.items()}
for k, v in sorted(fits.items(),
                   key=lambda x: x[1].aic):
    print(f"{k:12s}: AIC={v.aic:.1f}")
    print(f"BIC={v.bic:.1f}")
```

## Step 4: Best Model

```
# Step 4: Best Model Selection
# GJR-GARCH(1,1)-t typically wins
best = fits['GJR-t']
print(best.summary())

p = best.params
alpha = p['alpha[1]']
beta = p['beta[1]']
gamma = p.get('gamma[1]', 0)

print(f"\nalpha:      {alpha:.4f}")
print(f"beta:        {beta:.4f}")
print(f"gamma:         {gamma:.4f}")
print(f"Persist:      {alpha+beta:.4f}")
print(f"Leverage:     {gamma:.4f}")
print(f"      ({'Yes' if gamma > 0 else 'No'})")

# Half-life
import math
hl = math.log(2) / math.log(alpha + beta)
```

# BIST-100 Pipeline — Steps 5–6: Visualization and Forecast

*Plot conditional volatility and generate forward-looking estimates*

## Step 5: Visualization

```
# Step 5: Conditional Vol + VaR Plot
from scipy import stats
sig = best.conditional_volatility
sig_ann = sig * np.sqrt(252) / 100
nu = best.params['nu']
VaR99 = -(best.params['Const']/100
          + stats.t.ppf(0.01, nu) * sig / 100)

fig, (a1, a2) = plt.subplots(2, 1,
                             figsize=(12, 7), sharex=True)
log_ret.plot(ax=a1, c='#1A2A5E', lw=0.8,
             title='Log>Returns (%)')
(-VaR99 * 100).plot(ax=a2, c='#DC2626',
                   lw=1.2, label='99% VaR')
sig_ann.plot(ax=a2, c='#C9922A', lw=1.5,
            label='Annual Vol')
```

```
a2.legend()
a2.set_title('Conditional Vol and VaR')
```

## Step 6: Forecast

```
# Step 6: 30-Day Ahead Forecast
fc = best.forecast(horizon=30, reindex=False)
vol_fore = np.sqrt(fc.variance.iloc[-1]) \
            * np.sqrt(252) / 100

# Unconditional (long-run) volatility
p = best.params
unc_v = (p['omega'] / (1 - p['alpha[1]']
                     - p.get('gamma[1]', 0) / 2
                     - p['beta[1]']))**0.5
unc_ann = unc_v * np.sqrt(252) / 100

print("30-day vol forecast (ann.):")
print(vol_fore.round(4))
print(f"\nAvg forecast: {vol_fore.mean():.1%}")
print(f"Long-run vol: {unc_ann:.1%}")
print(f"Latest VaR99: "
      f"{abs(VaR99.iloc[-1])*100:.2f}%")
```

# Real-World: S&P 500 vs. Bitcoin Volatility

*Comparing volatility dynamics across asset classes*

## Multi-Asset Comparison

```
# Multi-asset GARCH comparison
tickers = {
    'S&P 500': '^GSPC',
    'BIST-100': 'XU100.IS',
    'Bitcoin': 'BTC-USD',
    'Gold': 'GC=F',
    'EUR/USD': 'EURUSD=X'
}

results_all = {}
for name, ticker in tickers.items():
    data = yf.download(ticker, start="2020-01-01",
                       end="2024-12-31", auto_adjust=True)['Close']
    ret = np.log(data/data.shift(1)).dropna() * 100
    m = arch_model(ret, vol='GARCH', p=1, o=1, q=1,
                   dist='t', mean='Constant')

    f = m.fit(dispatch='off')
    p = f.params
    results_all[name] = {
        'alpha': p['alpha[1:]'],
        'beta': p['beta[1:]'],
    }
```

# DCC-GARCH — Dynamic Conditional Correlation

*Engle (2002): Time-varying correlation for portfolio risk*

$$Q_t = (1-a-b) \cdot Q + a \cdot z_{t-1} z'_{t-1} + b \cdot Q_{t-1}$$

*Dynamic correlation matrix evolves as a function of past standardized residuals*

- Step 1: Fit univariate GARCH to each asset — obtain standardized residuals  $z_t$
- Step 2: Model the dynamic correlation matrix  $Q_t$  using the DCC equation above
- Step 3:  $R_t = \text{diag}(Q_t)^{-1/2} \cdot Q_t \cdot \text{diag}(Q_t)^{-1/2}$  — the time-varying correlation matrix
- Applications: Portfolio optimization, dynamic hedge ratios, cross-asset contagion, systemic risk measurement
- Limitation: Computationally expensive in high dimensions. Alternatives: DECO (Engle & Kelly, 2012), DCC-BEKK
- In this course: univariate GARCH focus; DCC is a reference topic for advanced applications.

# GARCH vs. Machine Learning

*Strengths and weaknesses of each approach for volatility forecasting*

| Criterion             | GARCH Approach                             | ML / Deep Learning                   |
|-----------------------|--------------------------------------------|--------------------------------------|
| Interpretability      | High — each parameter has economic meaning | Low — black box                      |
| Data requirement      | Low — 250 days is sufficient               | High — thousands of observations     |
| Regulatory compliance | Basel II/III accepted                      | No standard yet                      |
| Feature engineering   | Not needed — parametric model              | Required — features must be designed |
| Crisis performance    | Good — asymmetry modeled directly          | Risky — out-of-distribution problem  |
| Forecast quality      | Good — well-established                    | LSTM/HAR sometimes outperform        |
| Implementation        | arch library — easy                        | PyTorch/TF — more code               |

**Best use case:** Use GARCH for interpretability, VaR, regulatory compliance, ML for high-frequency, multifactor prediction. Many practitioners combine both — GARCH provides the variance series that feeds into ML models.

# Week 4 Summary

*What we learned — building blocks for Week 5*

## Volatility Clustering

Large changes follow large changes  
ARCH effect — conditional heteroskedasticity  
ARCH LM test as formal pre-diagnostic

## GARCH Family

GARCH(1,1):  $\omega + \alpha \varepsilon^2 + \beta \sigma^2$   
EGARCH: asymmetry via  $\gamma z$  term  
GJR-GARCH:  $I\{\varepsilon < 0\}$  leverage indicator

## Python Toolkit

arch library: `arch_model()`  
Normal/t/GED/skewt error distributions  
AIC/BIC model selection, LB test

## VaR and Risk

GARCH-VaR: dynamic, market-adaptive  
Backtesting via Kupiec LR test  
Basel II/III internal models approach

**Week 5: Financial Classification with Machine Learning — Logistic Regression, Random Forest, XGBoost**

# Week 4 Assignment

*Due: Before Week 5 class / Submit: .ipynb via LMS*

1

Apply the ARCH LM test to BIST-100 log-returns (lag = 5, 10, 20). Confirm the ARCH effect and interpret the squared residual ACF plot.

20 pts

2

Estimate four models: GARCH-Normal, GARCH-t, EGARCH-t, GJR-t. Build an AIC/BIC comparison table. Justify the winning model.

30 pts

3

Plot the winning model's annualized conditional volatility. Color-code three regimes (low/medium/high) and match them to real economic events.

25 pts

4

Calculate daily 99% and 95% GARCH-VaR for a 1M TL portfolio. Backtest over the last 252 days using the Kupiec LR test.

25 pts