

Why do we need GARCH models

GARCH MODELS IN PYTHON



Chelsea Yang

Data Science Instructor

Course overview

GARCH: Generalized AutoRegressive Conditional Heteroskedasticity

- Chapter 1: GARCH Model Fundamentals
- Chapter 2: GARCH Model Configuration
- Chapter 3: Model Performance Evaluation
- Chapter 4: GARCH in Action

What is volatility

- Describe the dispersion of financial asset returns over time
- Often computed as the standard deviation or variance of price returns
- The higher the volatility, the riskier a financial asset



How to compute volatility

- Step 1: Calculate returns as percentage of price changes

$$return = \frac{P_1 - P_0}{P_0}$$

- Step 2: Calculate the sample mean return

$$mean = \frac{\sum_{i=1}^n return_i}{n}$$

- Step 3: Calculate the sample standard deviation

$$volatility = \sqrt{\frac{\sum_{i=1}^n (return_i - mean)^2}{n - 1}} = \sqrt{variance}$$

Compute volatility in Python

Use pandas `pct_change()` method:

```
return_data = price_data.pct_change()
```

Use pandas `std()` method:

```
volatility = return_data.std()
```

Volatility conversion

- Convert to monthly volatility from daily:

(assume 21 trading days in a month)

$$\sigma_{monthly} = \sqrt{21} * \sigma_d$$

- Convert to annual volatility from daily:

(assume 252 trading days in a year)

$$\sigma_{annual} = \sqrt{252} * \sigma_d$$

The challenge of volatility modeling

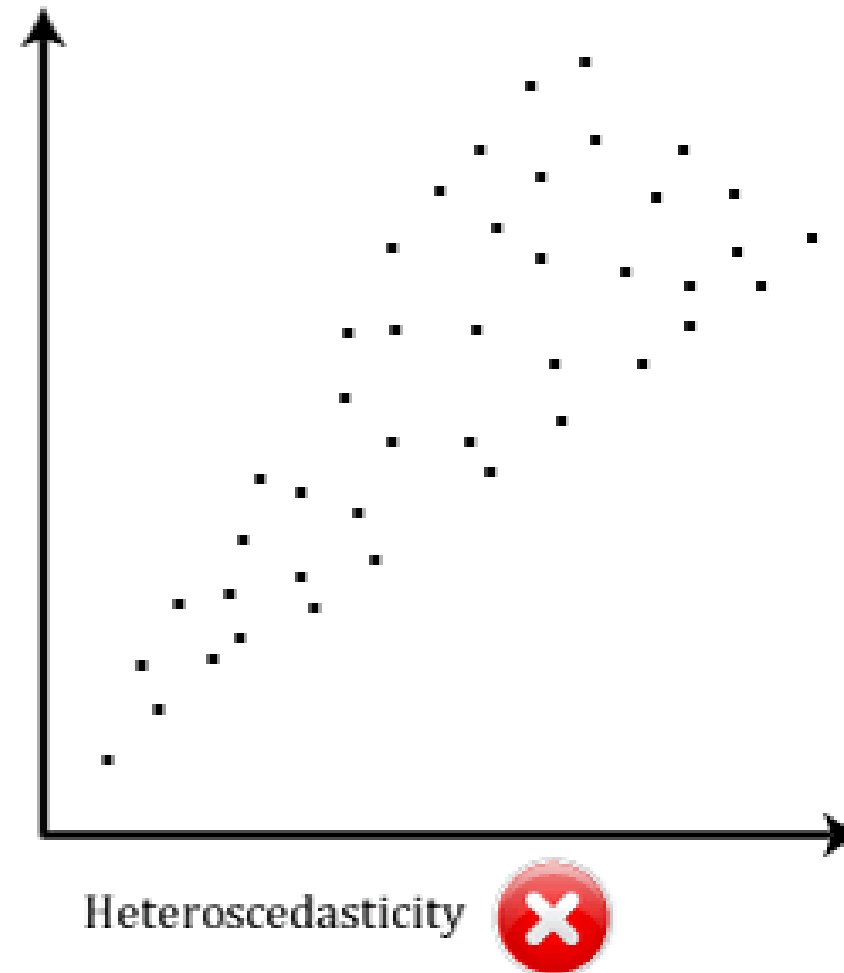
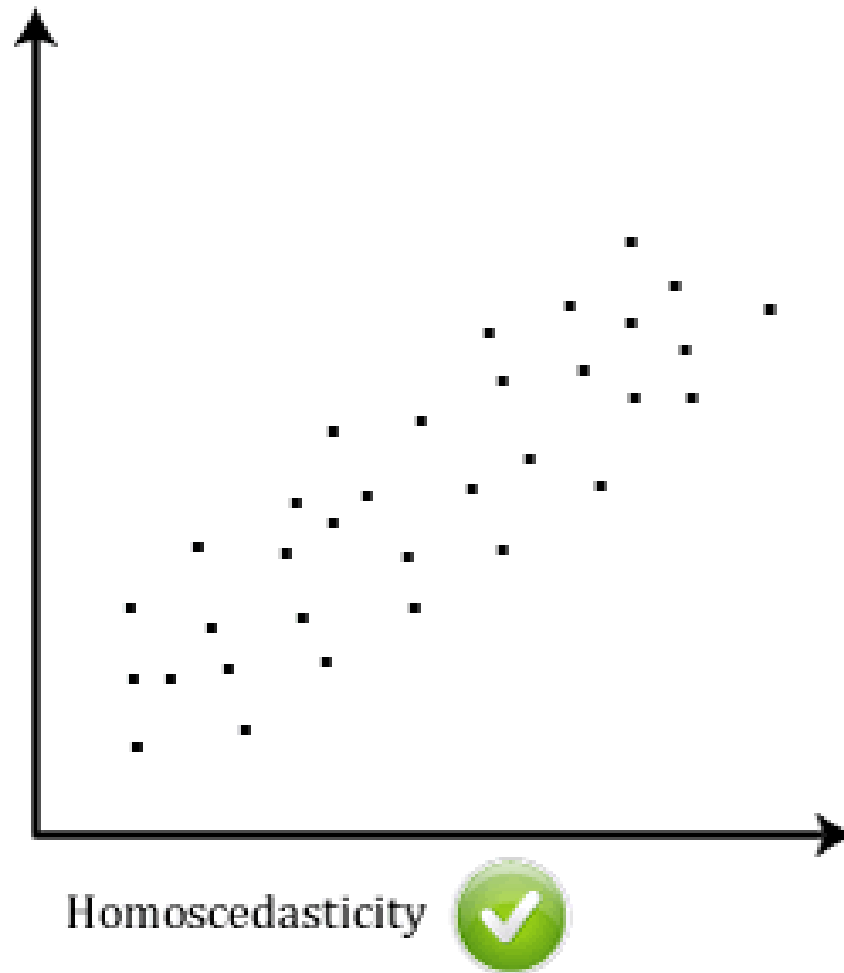
Heteroskedasticity:

- In ancient Greek: "different" (hetero) + "dispersion" (skedasis)
- A time series demonstrates varying volatility systematically over time



Detect heteroskedasticity

Homoskedasticity vs Heteroskedasticity



Volatility clustering

VIX historical prices:



Let's practice!
GARCH MODELS IN PYTHON

What are ARCH and GARCH

GARCH MODELS IN PYTHON

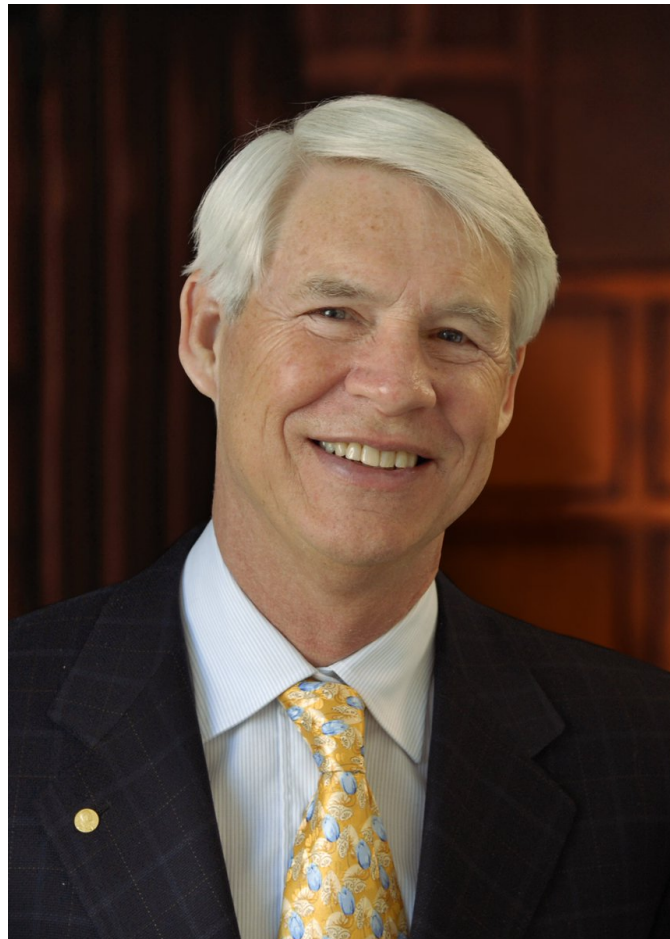


Chelsea Yang

Data Science Instructor

First came the ARCH

- Auto Regressive Conditional Heteroskedasticity
- Developed by Robert F. Engle (Nobel prize laureate 2003)



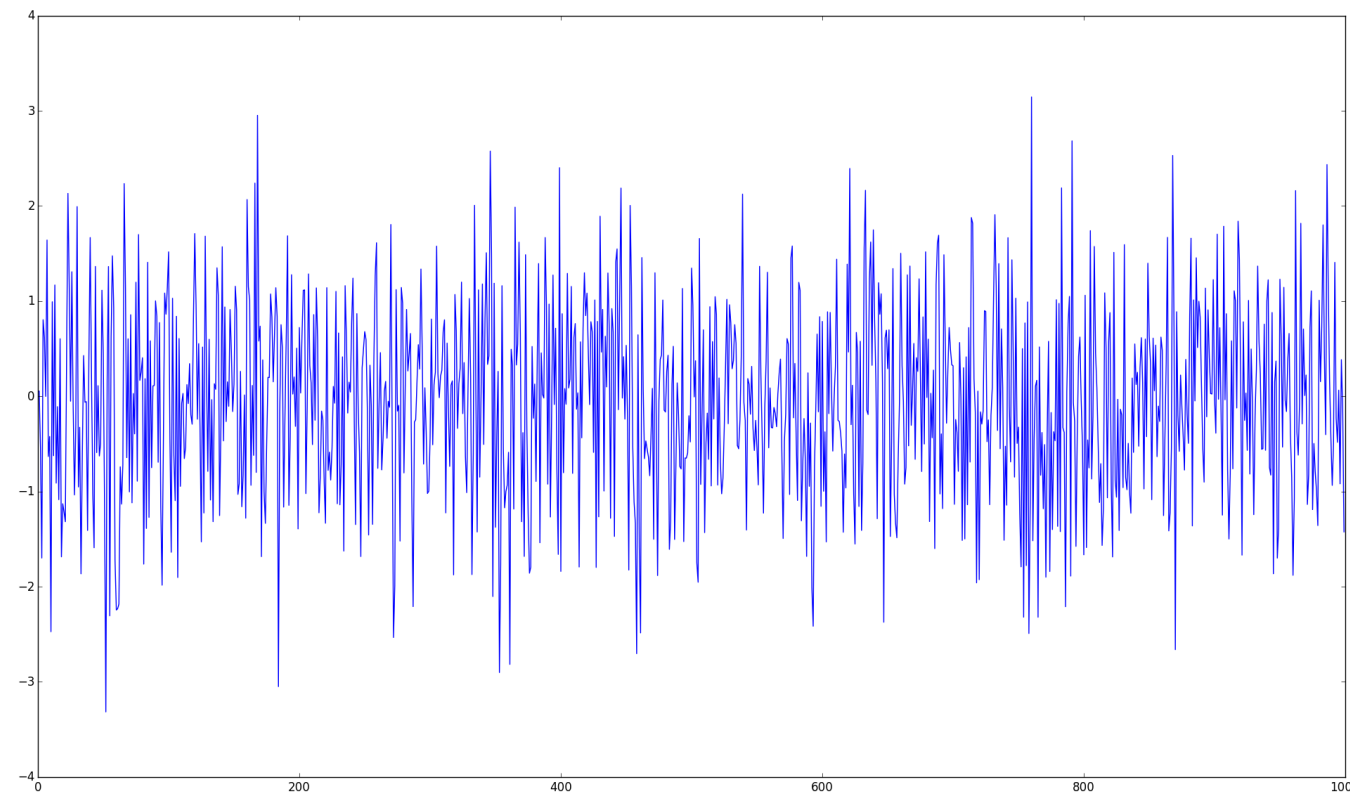
Then came the GARCH

- "Generalized" ARCH
- Developed by Tim Bollerslev (Robert F. Engle's student)



Related statistical terms

White noise (z): Uncorrelated random variables with a zero mean and a finite variance



Residual = predicted value - observed value

Model notations

Expected return:

$$\mu_t = \textit{Expected}[r_t | I(t - 1)]$$

Expected volatility:

$$\sigma^2 = \textit{Expected}[(r_t - \mu_t)^2 | I(t - 1)]$$

Residual (prediction error):

$$r_t = \mu_t + \epsilon_t$$

Volatility is related to the residuals:

$$\epsilon_t = \sigma_t * \zeta(\textit{WhiteNoise})$$

Model equations: ARCH

$$ARCH(p) : \sigma_t^2 = \omega + \sum_{i=1}^p \alpha_i \epsilon_{t-i}^2$$

$$ARCH(1) : \sigma_t^2 = \omega + \alpha \epsilon_{t-1}^2$$

Model equations: GARCH

$$GARCH(p, q) : \sigma_t^2 = \omega + \sum_{i=1}^p \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^q \beta_j \sigma_{t-j}^2$$

$$GARCH(1, 1) : \sigma_t^2 = \omega + \alpha \epsilon_{t-1}^2 + \beta \sigma_{t-1}^2$$

Model intuition

- Autoregressive: predict future behavior based on past behavior
- Volatility as a weighted average of past information



GARCH(1,1) parameter constraints

To make the GARCH(1,1) process realistic, it requires:

- All parameters are non-negative, so the variance cannot be negative.

$$\omega, \alpha, \beta \geq 0$$

- Model estimations are "mean-reverting" to the long-run variance.

$$\alpha + \beta < 1$$

long-run variance:

$$\omega / (1 - \alpha - \beta)$$

GARCH(1,1) parameter dynamics

- The larger the α , the bigger the immediate impact of the shock
- The larger the β , the longer the duration of the impact

Let's practice!
GARCH MODELS IN PYTHON

How to implement GARCH models in Python

GARCH MODELS IN PYTHON



Chelsea Yang

Data Science Instructor

Python "arch" package

```
from arch import arch_model
```



¹ Kevin Sheppard. (2019, March 28). bashtage/arch: Release 4.8.1 (Version 4.8.1). Zenodo.
<http://doi.org/10.5281/zenodo.2613877>

Workflow

Develop a GARCH model in three steps:

1. Specify the model
2. Fit the model
3. Make a forecast

Model specification

Model assumptions:

- Distribution: "normal" (default), "t", "skewt"
- Mean model: "constant" (default), "zero", "AR"
- Volatility model: "GARCH" (default), "ARCH", "EGARCH"

```
basic_gm = arch_model(sp_data['Return'], p = 1, q = 1,  
                      mean = 'constant', vol = 'GARCH', dist = 'normal')
```

Model fitting

Display model fitting output after every n iterations:

```
gm_result = gm_model.fit(update_freq = 4)
```

```
Iteration:      4,   Func. Count:      34,   Neg. LLF: 2783.005885607893
Iteration:      8,   Func. Count:      61,   Neg. LLF: 2771.9886612376513
Iteration:     12,   Func. Count:      85,   Neg. LLF: 2771.963828246998
Optimization terminated successfully.      (Exit mode 0)
      Current function value: 2771.9638282462456
      Iterations: 12
      Function evaluations: 85
      Gradient evaluations: 12
```

Turn off the display:

```
gm_result = gm_model.fit(disg = 'off')
```

Fitted results: parameters

Estimated by "maximum likelihood method"

```
print(gm_result.params)
```

```
mu          0.077239  
omega       0.039587  
alpha[1]    0.167963  
beta[1]     0.786467  
Name: params, dtype: float64
```

Fitted results: summary

```
print(gm_result.summary())
```

```

=====
                        Constant Mean - GARCH Model Results
=====
Dep. Variable:          Return      R-squared:          -0.001
Mean Model:             Constant Mean  Adj. R-squared:      -0.001
Vol Model:              GARCH         Log-Likelihood:     -2771.96
Distribution:           Normal        AIC:               5551.93
Method:                 Maximum Likelihood  BIC:              5574.95
                                     No. Observations:      2336
Date:                   Mon, Dec 02 2019  Df Residuals:      2332
Time:                   12:54:53          Df Model:           4
=====
                        Mean Model
=====

```

	coef	std err	t	P> t	95.0% Conf. Int.
mu	0.0772	1.445e-02	5.345	9.031e-08	[4.892e-02, 0.106]

```

=====
                        Volatility Model
=====

```

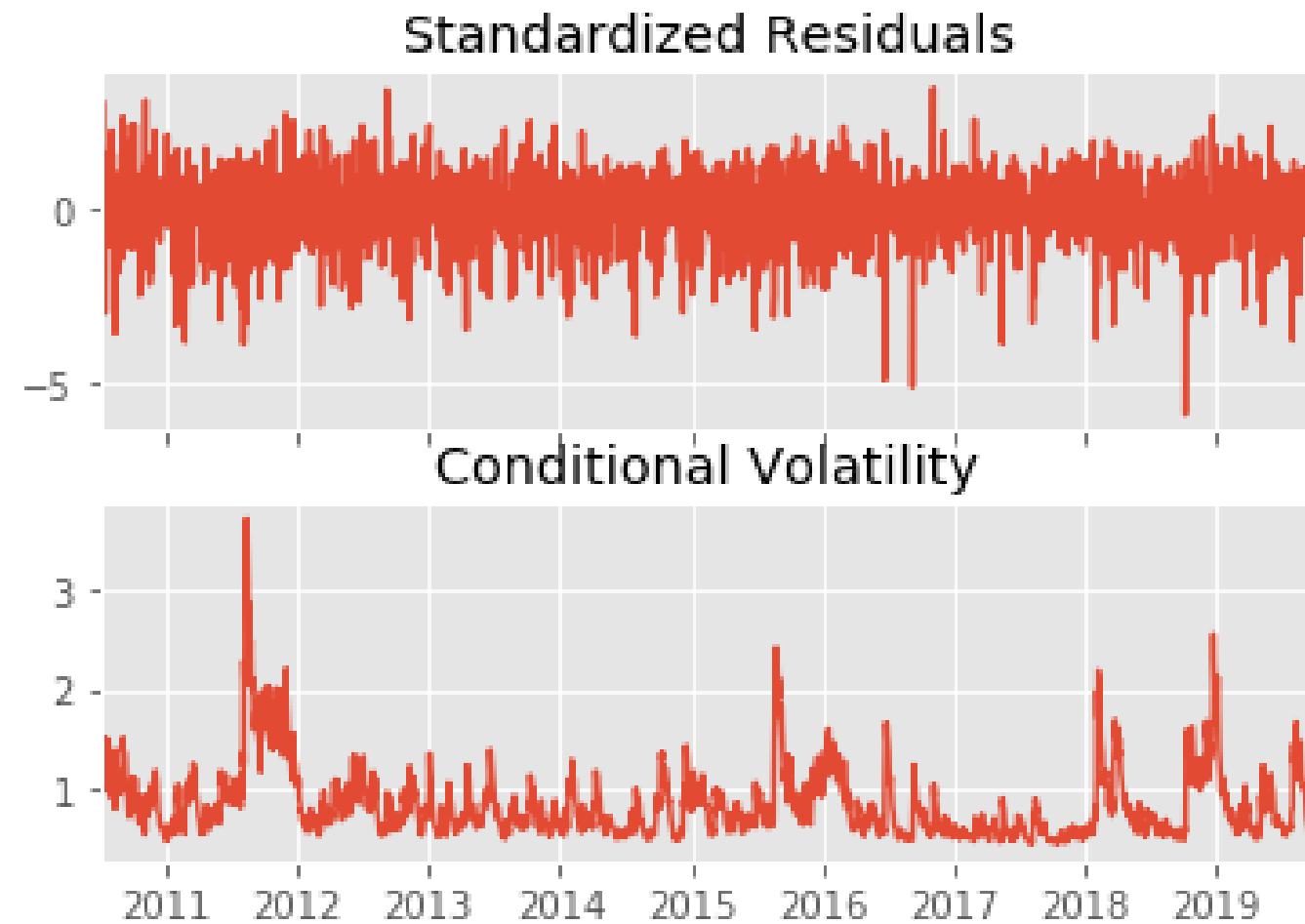
	coef	std err	t	P> t	95.0% Conf. Int.
omega	0.0396	9.181e-03	4.312	1.619e-05	[2.159e-02, 5.758e-02]
alpha[1]	0.1680	2.690e-02	6.243	4.284e-10	[0.115, 0.221]
beta[1]	0.7865	2.722e-02	28.897	1.303e-183	[0.733, 0.840]

```

=====
```

Fitted results: plots

```
gm_result.plot()
```



Model forecasting

```
# Make 5-period ahead forecast  
gm_forecast = gm_result.forecast(horizon = 5)
```

```
# Print out the last row of variance forecast  
print(gm_forecast.variance[-1:])
```

	h.1	h.2	h.3	h.4	h.5
Date					
2019-10-10	0.994079	0.988366	0.982913	0.977708	0.972741

h.1 in row "2019-10-10": 1-step ahead forecast made using data up to and including that date

Let's practice!
GARCH MODELS IN PYTHON