

Distribution assumptions

GARCH MODELS IN PYTHON



Chelsea Yang

Data Science Instructor

Why make assumptions

- Volatility is not directly observable
- GARCH model use residuals as volatility shocks

$$r_t = \mu_t + \epsilon_t$$

- Volatility is related to the residuals:

$$\epsilon_t = \sigma_t * \zeta(WhiteNoise)$$

Standardized residuals

- Residual = predicted return - mean return

$$residuals = \epsilon_t = r_t - \mu_t$$

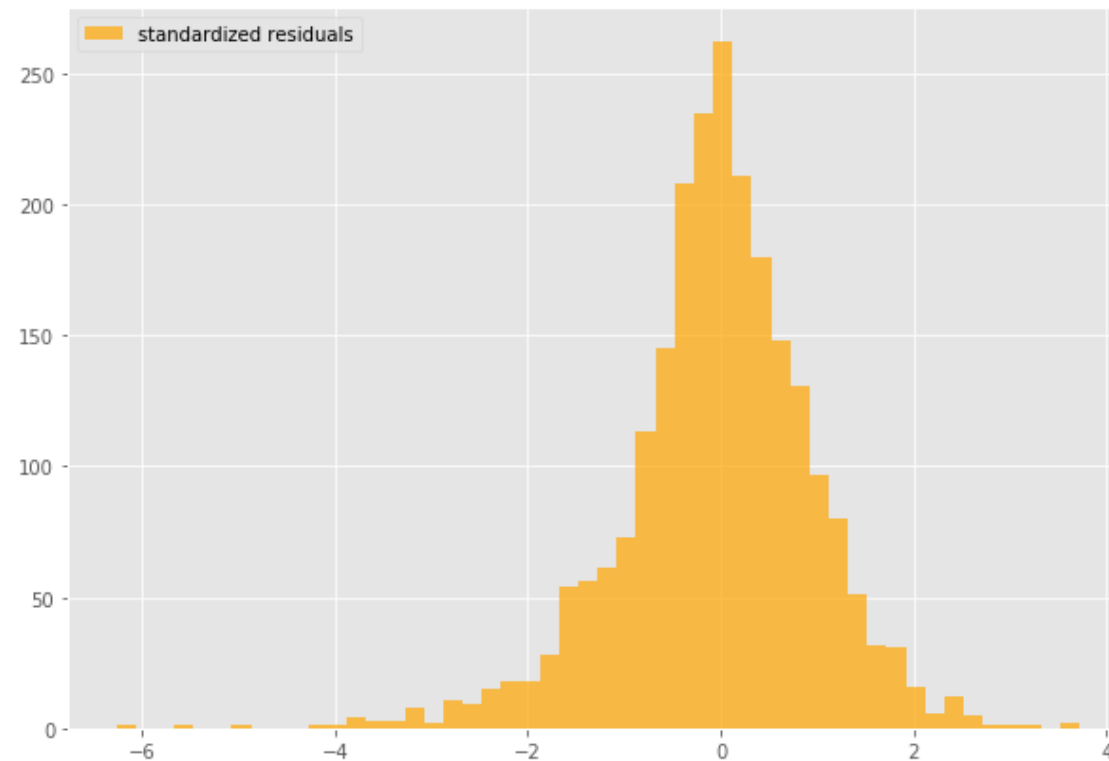
- Standardized residual = residual / return volatility

$$std\ Resid = \frac{\epsilon_t}{\sigma_t}$$

Residuals in GARCH

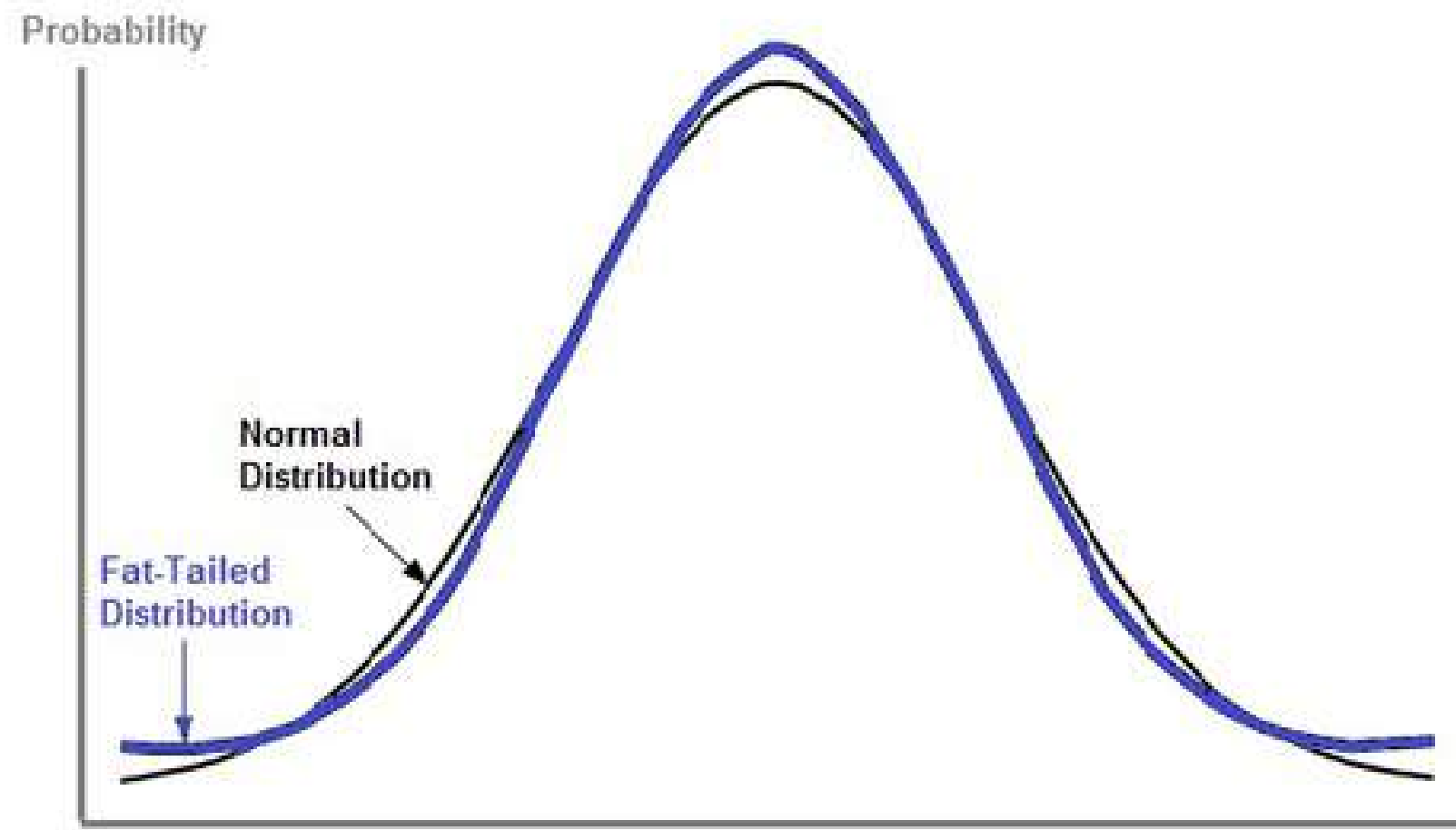
```
gm_std_resid = gm_result.resid / gm_result.conditional_volatility
```

```
plt.hist(gm_std_resid, facecolor = 'orange', label = 'standardized residuals')
```



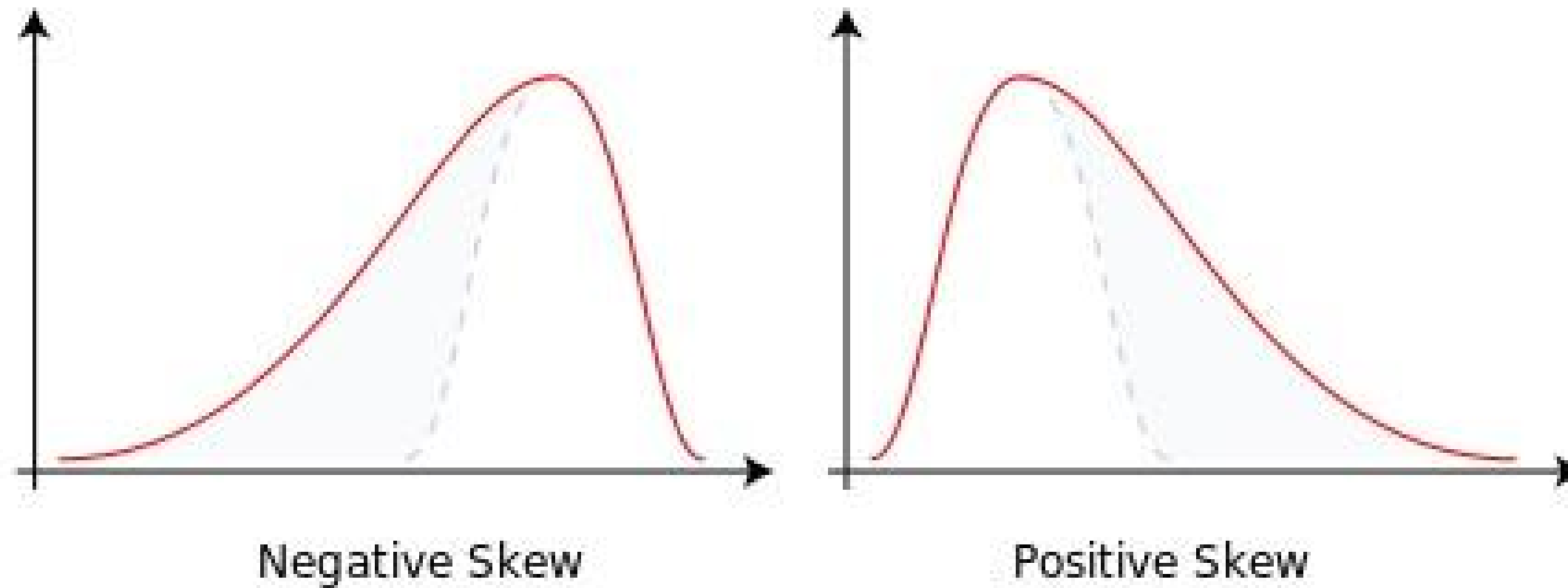
Fat tails

- Higher probability to observe large (positive or negative) returns than under a normal distribution

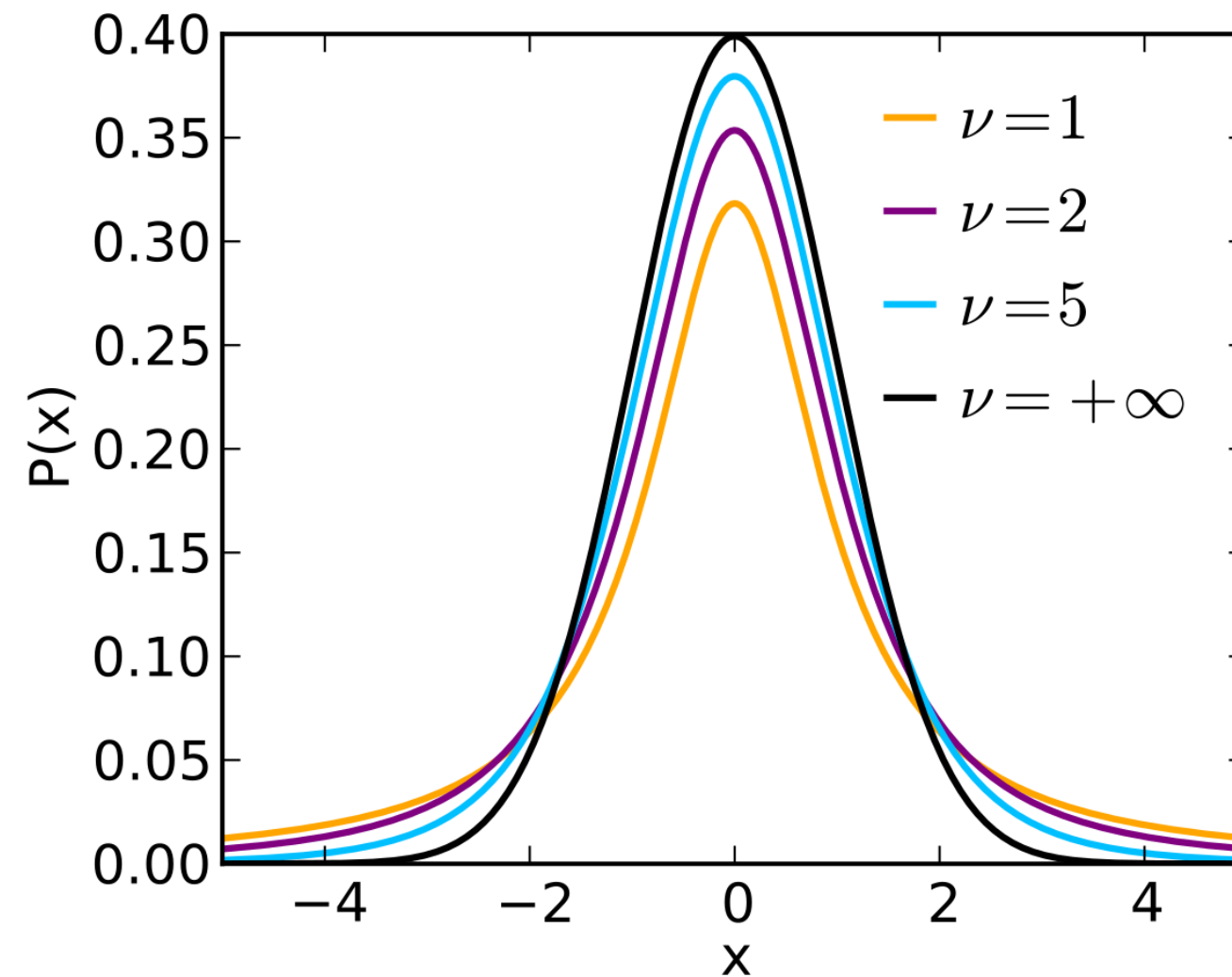


Skewness

- Measure of asymmetry of a probability distribution



Student's t-distribution



ν parameter of a Student's t-distribution indicates its shape

GARCH with t-distribution

```
arch_model(my_data, p = 1, q = 1,  
           mean = 'constant', vol = 'GARCH',  
           dist = 't')
```

	Distribution				
	coef	std err	t	P> t	95.0% Conf. Int.
nu	4.9249	0.507	9.709	2.766e-22	[3.931, 5.919]

GARCH with skewed t-distribution

```
arch_model(my_data, p = 1, q = 1,  
           mean = 'constant', vol = 'GARCH',  
           dist = 'skewt')
```

	Distribution				
	coef	std err	t	P> t	95.0% Conf. Int.
nu	5.2437	0.575	9.118	7.681e-20	[4.117, 6.371]
lambda	-0.0822	2.541e-02	-3.235	1.216e-03	[-0.132, -3.241e-02]

Let's practice!
GARCH MODELS IN PYTHON

Mean model specifications

GARCH MODELS IN PYTHON



Chelsea Yang

Data Science Instructor

Constant mean by default

- `constant` mean: generally works well with most financial return data

```
arch_model(my_data, p = 1, q = 1,  
           mean = 'constant', vol = 'GARCH')
```

```
Constant Mean - GARCH Model Results  
=====
```

Dep. Variable:	Return	R-squared:	-0.001
Mean Model:	<u>Constant Mean</u>	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:	Fri, Dec 20 2019	Df Residuals:	2332
Time:	05:26:46	Df Model:	4

```
Mean Model  
=====
```

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

Zero mean assumption

- `zero` mean: use when the mean has been modeled separately

```
arch_model(my_data, p = 1, q = 1,  
           mean = 'zero', vol = 'GARCH')
```

```
Zero Mean - GARCH Model Results  
=====
```

Dep. Variable:	Return	R-squared:	0.000
Mean Model:	<u>Zero Mean</u>	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	-2786.65
Distribution:	Normal	AIC:	5579.30
Method:	Maximum Likelihood	BIC:	5596.57
		No. Observations:	2336
Date:	Fri, Dec 20 2019	Df Residuals:	2333
Time:	05:36:28	Df Model:	3

```
-----
```

Autoregressive mean

- **AR** mean: model the mean as an autoregressive (AR) process

```
arch_model(my_data, p = 1, q = 1,  
           mean = 'AR', lags = 1, vol = 'GARCH')
```

```
AR - GARCH Model Results  
=====
```

Dep. Variable:	Return	R-squared:	0.001
Mean Model:	AR	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	-2690.07
Distribution:	Standardized Student's t	AIC:	5392.13
Method:	Maximum Likelihood	BIC:	5426.66
		No. Observations:	2335
Date:	Fri, Dec 20 2019	Df Residuals:	2329
Time:	05:39:58	Df Model:	6

```
Mean Model  
=====
```

	coef	std err	t	P> t	95.0% Conf. Int.
Const	0.0877	1.293e-02	6.783	1.181e-11	[6.234e-02, 0.113]
<u>Return[1]</u>	-0.0541	2.060e-02	-2.625	8.670e-03	[-9.444e-02, -1.369e-02]

Let's practice!
GARCH MODELS IN PYTHON

Volatility models for asymmetric shocks

GARCH MODELS IN PYTHON

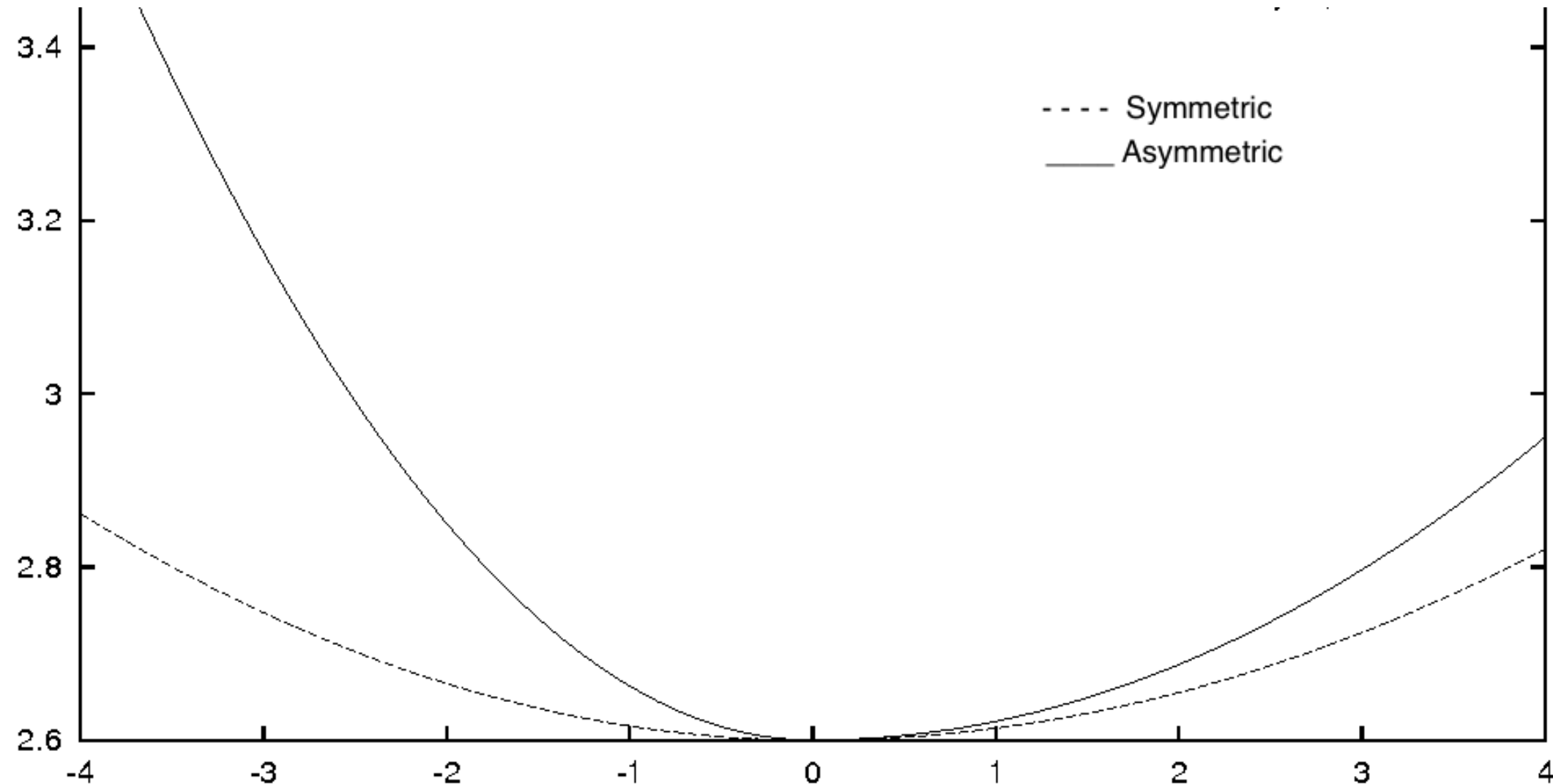


Chelsea Yang

Data Science Instructor

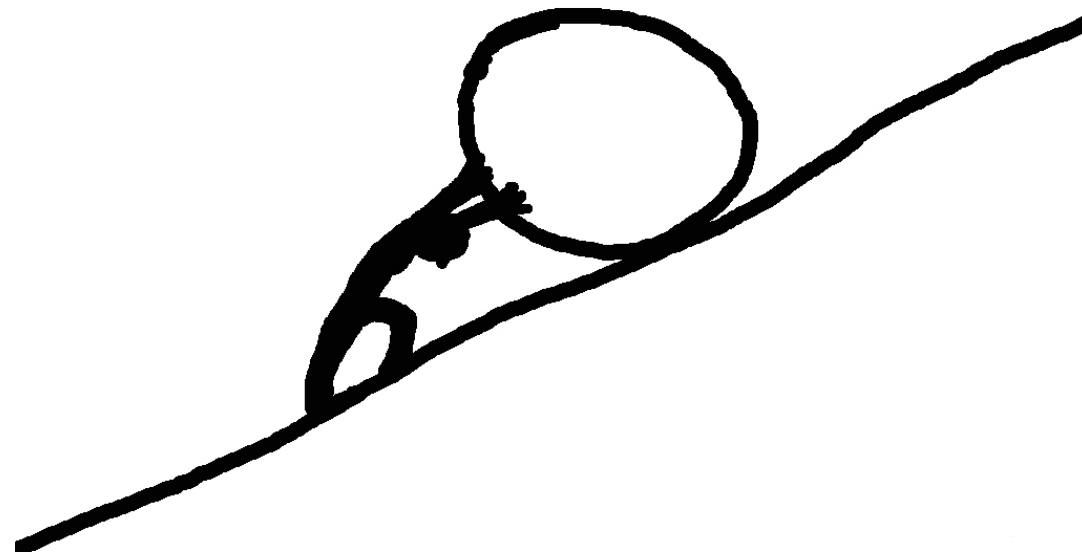
Asymmetric shocks in financial data

News impact curve:



Leverage effect

- Debt-equity Ratio = Debt / Equity
- Stock price goes down, debt-equity ratio goes up
- Riskier!



GJR-GARCH

$$\sigma_t^2 = \omega + (\alpha + \gamma I_{t-1})\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

$$I_{t-1} := \begin{cases} 0 & \text{if } r_{t-1} \geq \mu \\ 1 & \text{if } r_{t-1} < \mu \end{cases}$$

GJR-GARCH in Python

```
arch_model(my_data, p = 1, q = 1, o = 1,  
           mean = 'constant', vol = 'GARCH')
```

```
=====
                        Constant Mean - GJR-GARCH Model Results
=====
Dep. Variable:                Return    R-squared:                -0.000
Mean Model:                  Constant Mean    Adj. R-squared:           -0.000
Vol Model:                   GJR-GARCH      Log-Likelihood:          -2641.12
Distribution:                Standardized Student's t    AIC:                    5294.23
Method:                      Maximum Likelihood    BIC:                    5328.77
                                     No. Observations:                2336
Date:                        Tue, Dec 10 2019    Df Residuals:            2330
Time:                        11:19:41    Df Model:                 6
                                     Mean Model
=====
                        coef    std err          t      P>|t|      95.0% Conf. Int.
-----
mu                0.0554  1.227e-02     4.521  6.163e-06  [3.141e-02,7.949e-02]
                                     Volatility Model
=====
                        coef    std err          t      P>|t|      95.0% Conf. Int.
-----
omega             0.0298  5.609e-03     5.317  1.054e-07  [1.883e-02,4.082e-02]
alpha[1]          0.0000  2.338e-02     0.000      1.000  [-4.583e-02,4.583e-02]
gamma[1]         0.3267  4.852e-02     6.733  1.663e-11  [ 0.232,  0.422]
beta[1]           0.8121  2.257e-02    35.978  1.835e-283  [ 0.768,  0.856]
```

EGARCH

- A popular option to model asymmetric shocks
- Exponential GARCH
- Add a conditional component to model the asymmetry in shocks similar to the GJR-GARCH
- No non-negative constraints on alpha, beta so it runs faster

EGARCH in Python

```
arch_model(my_data, p = 1, q = 1, o = 1,  
           mean = 'constant', vol = 'EGARCH')
```

```
=====
                        Constant Mean - EGARCH Model Results
=====
Dep. Variable:          Return      R-squared:          -0.000
Mean Model:             Constant Mean  Adj. R-squared:         -0.000
Vol Model:              EGARCH      Log-Likelihood:       -2628.40
Distribution:           Standardized Student's t    AIC:             5268.79
Method:                Maximum Likelihood          BIC:             5303.33
                                           No. Observations:      2336
Date:                  Tue, Dec 10 2019            Df Residuals:         2330
Time:                  11:19:42                    Df Model:              6
                                           Mean Model
=====
                        coef      std err          t      P>|t|      95.0% Conf. Int.
-----
mu                0.0493   9.578e-03      5.146   2.663e-07   [3.051e-02, 6.806e-02]
                        Volatility Model
=====
                        coef      std err          t      P>|t|      95.0% Conf. Int.
-----
omega            -0.0202   7.350e-03     -2.743   6.094e-03   [-3.457e-02, -5.753e-03]
alpha[1]         0.1707   2.279e-02      7.490   6.874e-14           [ 0.126,  0.215]
gamma[1]        -0.2360   2.598e-02     -9.087   1.019e-19           [-0.287, -0.185]
beta[1]          0.9547   9.191e-03    103.869   0.000           [ 0.937,  0.973]
```

Which model to use

GJR-GARCH or EGARCH?

Which model is better depends on the data



Let's practice!
GARCH MODELS IN PYTHON

GARCH rolling window forecast

GARCH MODELS IN PYTHON



Chelsea Yang

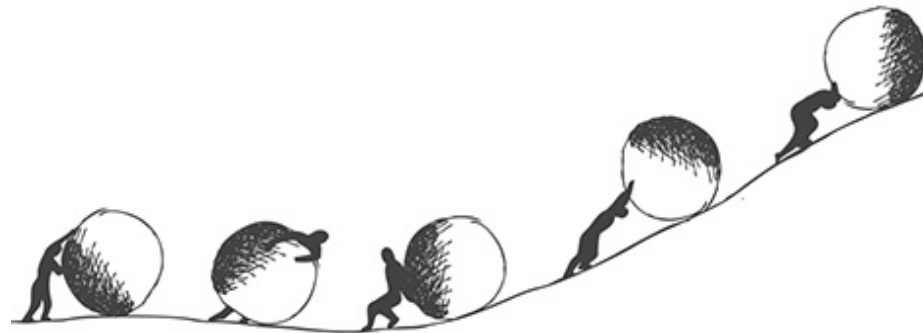
Data Science Instructor

Rolling window for out-of-sample forecast

An exciting part of financial modeling: **predict the unknown**



Rolling window forecast: repeatedly perform model fitting and forecast as time rolls forward



Expanding window forecast

Continuously add new data points to the sample



Motivations of rolling window forecast

- Avoid lookback bias
- Less subject to overfitting
- Adapt forecast to new observations

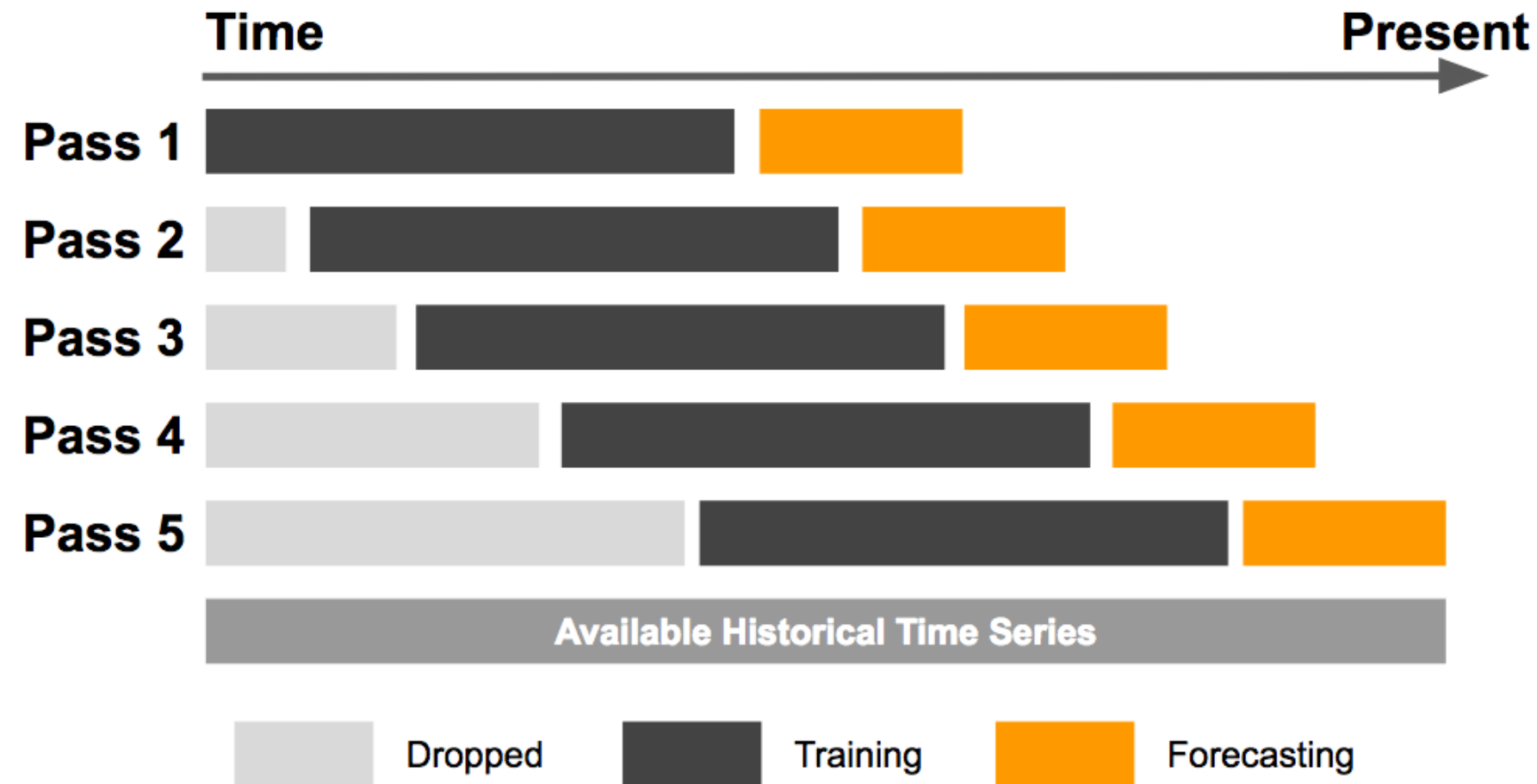
Implement expanding window forecast

Expanding window forecast:

```
for i in range(120):  
    gm_result = basic_gm.fit(first_obs = start_loc,  
                             last_obs = i + end_loc, disp = 'off')  
    temp_result = gm_result.forecast(horizon = 1).variance
```

Fixed rolling window forecast

New data points are added while old ones are dropped from the sample



Implement fixed rolling window forecast

Fixed rolling window forecast:

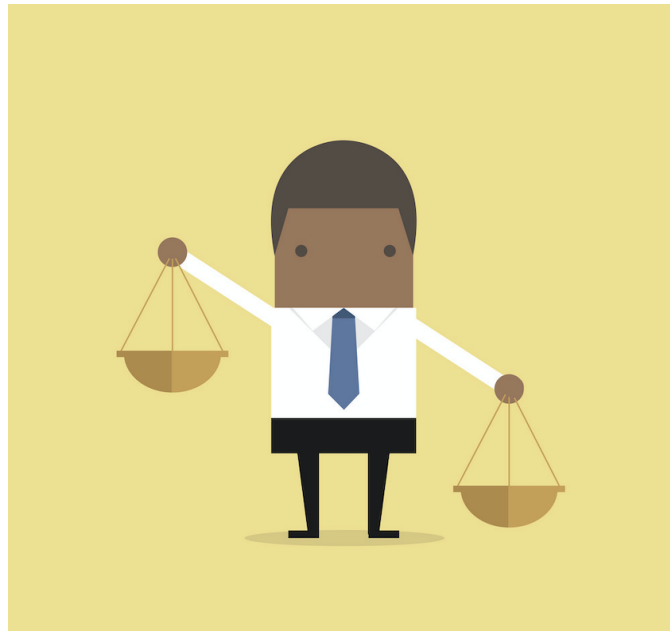
```
for i in range(120):  
    # Specify rolling window range for model fitting  
    gm_result = basic_gm.fit(first_obs = i + start_loc,  
                             last_obs = i + end_loc, disp = 'off')  
    temp_result = gm_result.forecast(horizon = 1).variance
```

How to determine window size

Usually determined on a case-by-case basis

- Too wide window size: include obsolete data that may lead to high bias
- Too narrow window size: exclude relevant data that may lead to higher variance

The optimal window size: trade-off to balance bias and variance



Let's practice!
GARCH MODELS IN PYTHON