02_arch_garch_models

September 29, 2021

1 ARCH/GARCH Volatility Forecasting

The development of a volatility model for an asset-return series consists of four steps: 1. Build an ARMA time series model for the financial time series based on the serial dependence revealed by the ACF and PACF. 2. Test the residuals of the model for ARCH/GARCH effects, again relying on the ACF and PACF for the series of the squared residual. 3. Specify a volatility model if serial correlation effects are significant, and jointly estimate the mean and volatility equations. 4. Check the fitted model carefully and refine it if necessary

When applying volatility forecasting to return series, the serial dependence may be limited so that a constant mean may be used instead of an ARMA model.

The `arch` library provides several options to estimate volatility-forecasting models. It offers several options to model the expected mean, including a constant mean, the AR(p) model discussed in the section on univariate time series models above as well as more recent heterogeneous autoregressive processes (HAR) that use daily (1 day), weekly (5 days), and monthly (22 days) lags to capture the trading frequencies of short-, medium-, and long-term investors.

The mean models can be jointly defined and estimated with several conditional heteroskedasticity models that include, in addition to ARCH and GARCH, the exponential GARCH (EGARCH) model, which allows for asymmetric effects between positive and negative returns and the heterogeneous ARCH (HARCH) model, which complements the HAR mean model.

1.1 Imports & Settings

[1]:
```
%matplotlib inline

import os
import sys
import warnings
from datetime import date

import pandas as pd
import pandas_datareader.data as web

import numpy as np
from numpy.linalg import LinAlgError

import statsmodels.api as sm
```

```
import statsmodels.tsa.api as tsa
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import acf, q_stat, adfuller

from sklearn.metrics import mean_squared_error

from scipy.stats import probplot, moment

from arch import arch_model
from arch.univariate import ConstantMean, GARCH, Normal

import matplotlib.pyplot as plt
import matplotlib as mpl
```

[2]:
```
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

1.2 Correlogram Plot

[3]:
```
def plot_correlogram(x, lags=None, title=None):
    lags = min(10, int(len(x)/5)) if lags is None else lags
    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 8))
    x.plot(ax=axes[0][0])
    q_p = np.max(q_stat(acf(x, nlags=lags), len(x))[1])
    stats = f'Q-Stat: {np.max(q_p):>8.2f}\nADF: {adfuller(x)[1]:>11.2f}'
    axes[0][0].text(x=.02, y=.85, s=stats, transform=axes[0][0].transAxes)
    probplot(x, plot=axes[0][1])
    mean, var, skew, kurtosis = moment(x, moment=[1, 2, 3, 4])
    s = f'Mean: {mean:>12.2f}\nSD: {np.sqrt(var):>16.2f}\nSkew: {skew:12.
 ↪2f}\nKurtosis:{kurtosis:9.2f}'
    axes[0][1].text(x=.02, y=.75, s=s, transform=axes[0][1].transAxes)
    plot_acf(x=x, lags=lags, zero=False, ax=axes[1][0])
    plot_pacf(x, lags=lags, zero=False, ax=axes[1][1])
    axes[1][0].set_xlabel('Lag')
    axes[1][1].set_xlabel('Lag')
    fig.suptitle(title, fontsize=20)
    fig.tight_layout()
    fig.subplots_adjust(top=.9)
```

1.3 Download NASDAQ Index Data

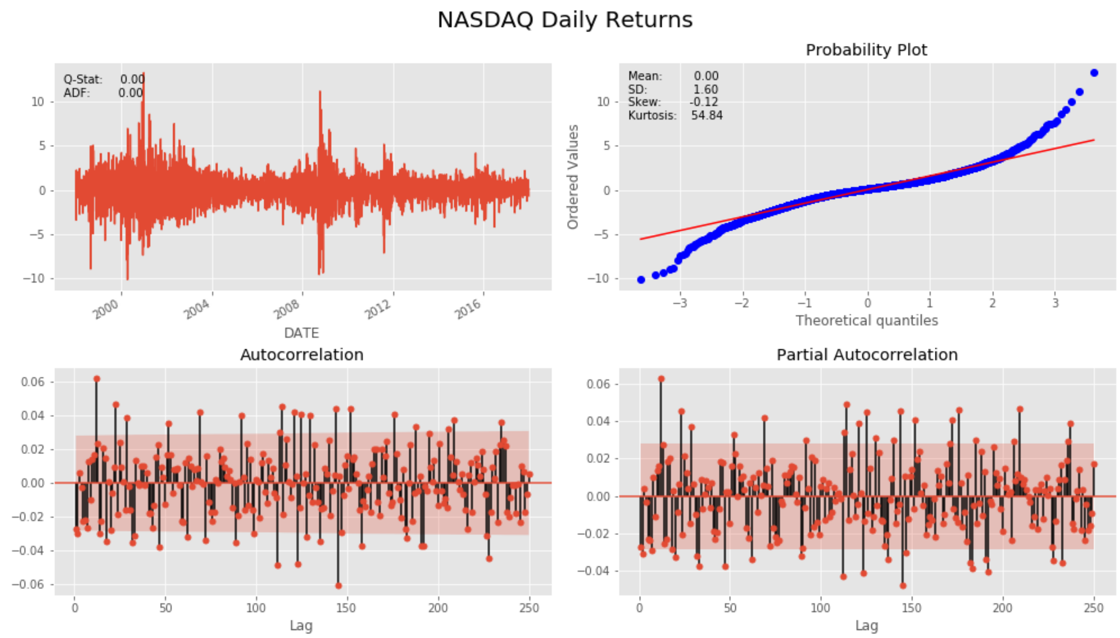
We will use daily NASDAQ returns from 1998-2017 to demonstrate the usage of a GARCH model

[4]:
```
nasdaq = web.DataReader('NASDAQCOM', 'fred', '1998', '2017-12-31').squeeze()
nasdaq_returns = np.log(nasdaq).diff().dropna().mul(100) # rescale to faciliate␣
 ↪optimization
```

1.4 Explore Returns and Volatility

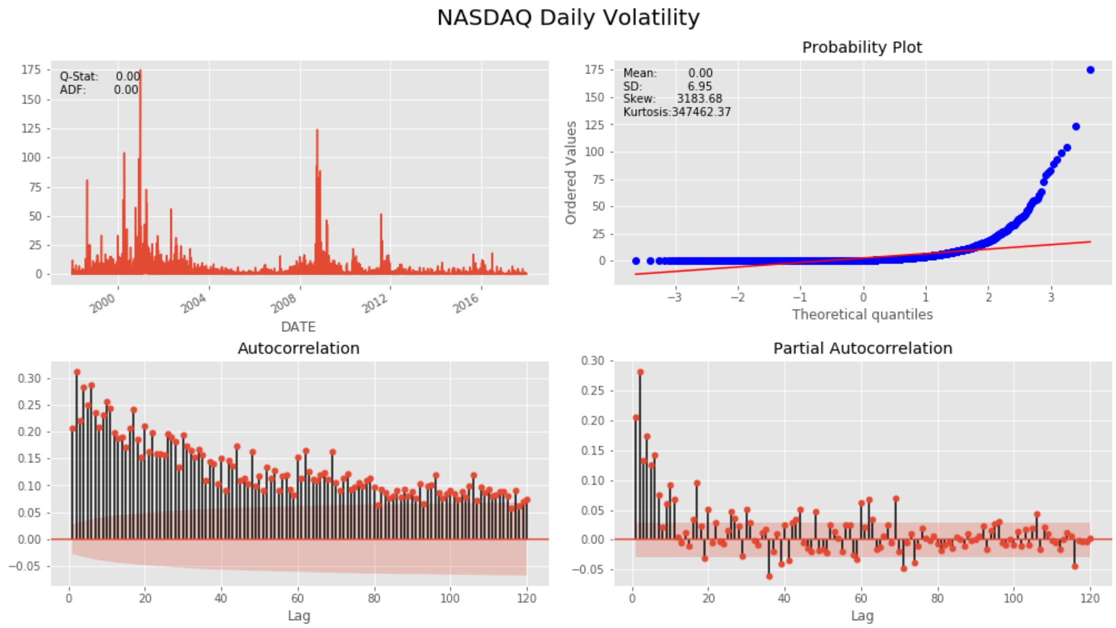
The rescaled daily return series exhibits only limited autocorrelation, but the squared deviations from the mean do have substantial memory reflected in the slowly-decaying ACF and the PACF high for the first two and cutting off only after the first six lags:

[5]:
```
plot_correlogram(nasdaq_returns, lags=250, title='NASDAQ Daily Returns')
```



The function plot_correlogram produces the following output:

[6]:
```
plot_correlogram(nasdaq_returns.sub(nasdaq_returns.mean()).pow(2), lags=120,␣
    ↪title='NASDAQ Daily Volatility')
```



1.5 Model Selection: rolling out-of-sample forecasts

Hence, we can estimate a GARCH model to capture the linear relationship of past volatilities. We will use rolling 10-year windows to estimate a GARCH(p, q) model with p and q ranging from 1-4 to generate 1-step out-of-sample forecasts.

We then compare the RMSE of the predicted volatility relative to the actual squared deviation of the return from its mean to identify the most predictive model. We are using winsorized data to limit the impact of extreme return values reflected in the very high positive skew of the volatility

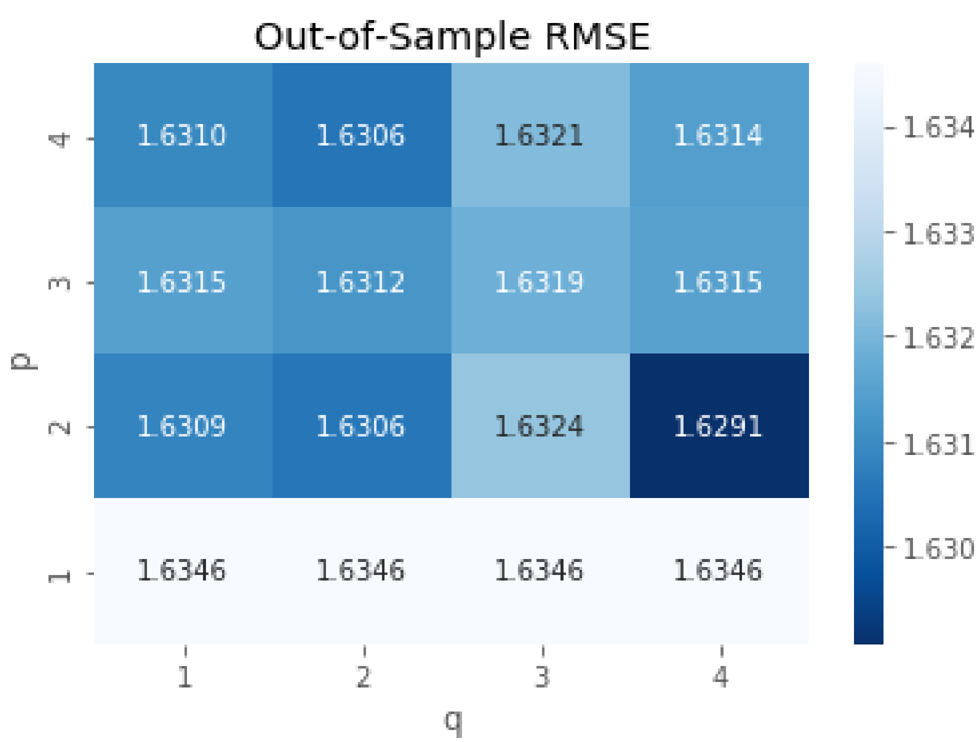
[7]:
```
trainsize = 10 * 252  # 10 years
data = nasdaq_returns.clip(lower=nasdaq_returns.quantile(.05),
                           upper=nasdaq_returns.quantile(.95))
T = len(nasdaq_returns)
test_results = {}
for p in range(1, 5):
    for q in range(1, 5):
        print(f'{p} | {q}')
        result = []
        for s, t in enumerate(range(trainsize, T-1)):
            train_set = data.iloc[s: t]
            test_set = data.iloc[t+1]  # 1-step ahead forecast
            model = arch_model(y=train_set, p=p, q=q).fit(disp='off')
            forecast = model.forecast(horizon=1)
            mu = forecast.mean.iloc[-1, 0]
            var = forecast.variance.iloc[-1, 0]
            result.append([(test_set-mu)**2, var])
```

```
        df = pd.DataFrame(result, columns=['y_true', 'y_pred'])
        test_results[(p, q)] = np.sqrt(mean_squared_error(df.y_true, df.y_pred))
```

```
1 | 1
1 | 2
1 | 3
1 | 4
2 | 1
2 | 2
2 | 3
2 | 4
3 | 1
3 | 2
3 | 3
3 | 4
4 | 1
4 | 2
4 | 3
4 | 4
```

[8]:

```
s = pd.Series(test_results)
s.index.names = ['p', 'q']
s = s.unstack().sort_index(ascending=False)
sns.heatmap(s, cmap='Blues_r', annot=True, fmt='.4f')
plt.title('Out-of-Sample RMSE');
```



1.6 Estimate GARCH(2, 2) Model

The GARCH(2, 2) model achieves the lowest RMSE (same value as GARCH(4, 2) but with fewer parameters), so we go ahead and estimate this model to inspect the summary.

The output shows the maximized log-likelihood as well as the AIC and BIC criteria that are commonly minimized when selecting models based on in-sample performance (see Chapter 7, Linear Models). It also displays the result for the mean model, which in this case is just a constant estimate, as well as the GARCH parameters for the constant omega, the AR parameters, , and the MA parameters, , all of which are statistically significant:

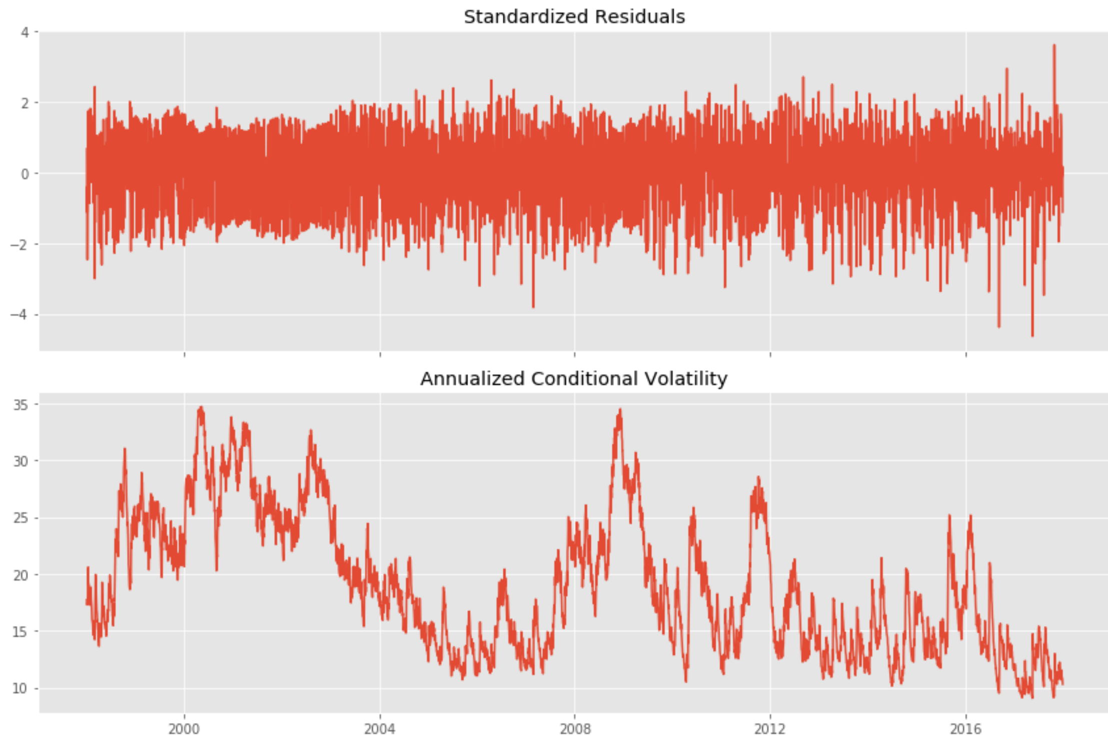
[9]:
```
am = ConstantMean(nasdaq_returns.clip(lower=nasdaq_returns.quantile(.05),
                                      upper=nasdaq_returns.quantile(.95)))
am.volatility = GARCH(2, 0, 2)
am.distribution = Normal()
model = am.fit(update_freq=5)
print(model.summary())
```

```
Iteration:      5,   Func. Count:     50,   Neg. LLF: 7485.702807539992
Iteration:     10,   Func. Count:     95,   Neg. LLF: 7484.0554748785635
Iteration:     15,   Func. Count:    136,   Neg. LLF: 7484.022927167949
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 7484.02292716963
            Iterations: 15
            Function evaluations: 136
            Gradient evaluations: 15
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                NASDAQCOM   R-squared:                      -0.001
Mean Model:               Constant Mean   Adj. R-squared:                 -0.001
Vol Model:                        GARCH   Log-Likelihood:               -7484.02
Distribution:                    Normal   AIC:                           14980.0
Method:              Maximum Likelihood   BIC:                           15019.0
                                          No. Observations:                 4852
Date:                  Wed, Oct 31 2018   Df Residuals:                     4846
Time:                          17:46:35   Df Model:                            6
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0521  1.491e-02      3.491  4.804e-04 [2.284e-02,8.130e-02]
                              Volatility Model
=============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
-----------------------------------------------------------------------------
omega          0.0196  8.287e-03      2.365  1.804e-02  [3.354e-03,3.584e-02]
```

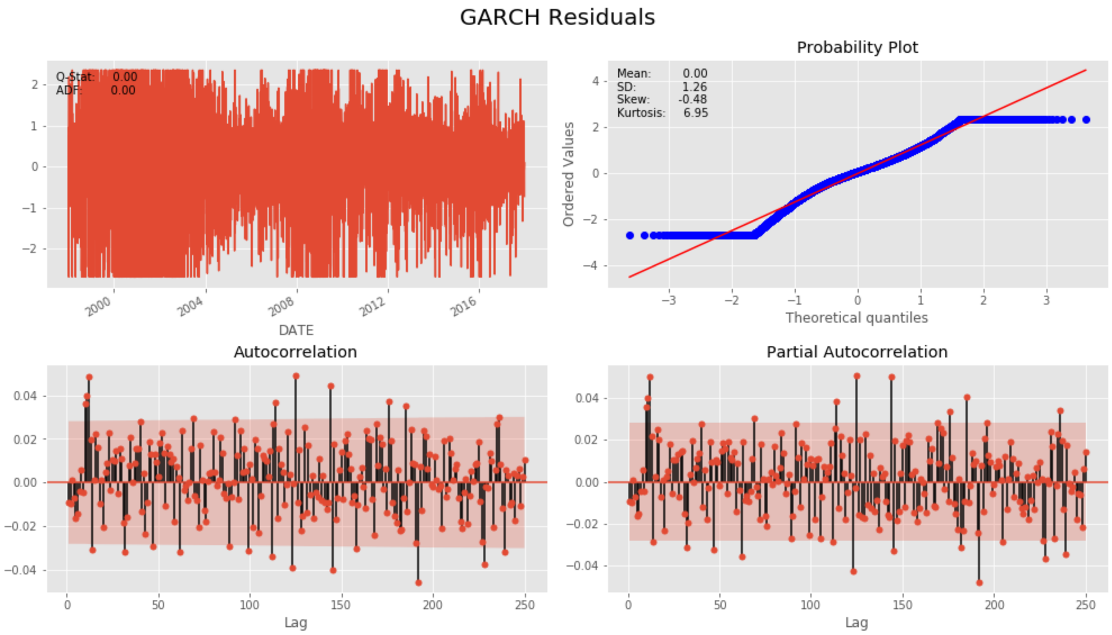
```
alpha[1]        0.0247  1.470e-02      1.678  9.340e-02 [-4.148e-03,5.346e-02]
alpha[2]        0.0627  2.196e-02      2.853  4.324e-03    [1.962e-02,  0.106]
beta[1]         0.5648      0.181      3.120  1.806e-03       [  0.210,  0.920]
beta[2]         0.3337      0.180      1.853  6.393e-02   [-1.932e-02,  0.687]
============================================================================

Covariance estimator: robust
```

1.6.1 Check Residuals

[12]:
```python
fig = model.plot(annualize='D')
fig.set_size_inches(12, 8)
fig.tight_layout();
```



[13]:
```python
plot_correlogram(model.resid.dropna(), lags=250, title='GARCH Residuals')
```



GARCH Residuals
Q-Stat: 0.00
ADF: 0.00
DATE
Probability Plot
Mean: 0.00
SD: 1.26
Skew: -0.48
Kurtosis: 6.95
Ordered Values
Theoretical quantiles
Autocorrelation
Lag
Partial Autocorrelation
Lag


```
[ ]:
```