时序分析(8) -- GARCH(p,q)模型

如无特殊说明，本系列文章中的数据将使用2012~2017年，分别代表国内股票、香港股票、国内债卷和国内货币的四个指数数据。

上篇文章我们探讨了ARCH模型对时序数据的波动性进行建模和预测，本篇文章介绍GARCH模型。 首先我们介绍GARCH模型的基本概念：

Generalized Autoregressive Conditionally Heteroskedastic Models - GARCH(p,q)

简单来说，GARCH模型就是ARMA模型应用在时序的方差上，它包含一个自回归项和一个移动平均项。

如果时序数据$\{y_t\}$可以表示为

$$y_t = \sigma_t w_t$$

其中$\{w_t\}$是高斯白噪声，均值为0，方差为1，这里σ_t为

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i y_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

让我们先从简单做起，考虑GARCH(1,1)模型

$$y_t = \sigma_t w_t$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

注意：这里$\alpha_1 + \beta_1 < 1$，否则时序将会不稳定。

In [1]:

```
import warnings
warnings.simplefilter('ignore')
```

1. 导入python包

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from fintechtools.backtest import *
from fintechtools.datasource import *
#from fintechtools.SimuMultiTest import *
#from lib.portfolio import DailySimulator
#from lib.experiment import Experiment

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from arch import arch_model
#sns.set_context("talk")
import matplotlib
import matplotlib as mpl
from matplotlib.ticker import FuncFormatter
mpl.style.use('classic')

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['font.serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
import seaborn as sns
sns.set_style("whitegrid",{"font.sans-serif":['simhei', 'Arial']})
sns.set_context("talk")

#zhfont1 = matplotlib.font_manager.FontProperties(fname='C:\Users\ktwc37\Documents\ZNTG\notebooks\SI


%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

2. 读入数据

In [3]:

```
start = '2012-01-01'
end = '2017-02-05'
```

In [4]:

```
indexs = pd.read_excel('./data/华夏指数.xlsx')
indexs_pv = indexs.pivot_table(index='日期', columns='简称', values='收盘价(元)')
indexs_pv.index = pd.to_datetime(indexs_pv.index, unit='d')
```

In [5]:

```
indexs_pv.columns = ['国内债券', '国内股票', '香港股票', '国内货币']
indexs_pv = indexs_pv[['国内债券', '国内股票', '国内货币', '香港股票']]
indexs_pv.fillna(axis=0,method='bfill',inplace=True)
indexs_sub = indexs_pv.loc[start:end,]
```

国内债卷：中债综合财富(总值)指数
国内股票：中证全指
香港股票：恒生指数
国内货币：货币基金

In [6]:

```
indexs_sub.head()
```

Out[6]:

日期	国内债券	国内股票	国内货币	香港股票
2012-01-04	141.5160	2571.951	1166.7726	18727.31
2012-01-05	141.5501	2513.699	1166.9696	18813.41
2012-01-06	141.7277	2527.247	1167.1185	18593.06
2012-01-09	141.8669	2619.638	1167.5058	18865.72
2012-01-10	142.0118	2713.529	1167.6330	19004.28

In [7]:

```
indexs_logret = indexs_sub.apply(log_return).dropna()
```

In [8]:

```
indexs_logret.head()
```

Out[8]:

日期	国内债券	国内股票	国内货币	香港股票
2012-01-05	0.000241	-0.022909	0.000169	0.004587
2012-01-06	0.001254	0.005375	0.000128	-0.011782
2012-01-09	0.000982	0.035906	0.000332	0.014558
2012-01-10	0.001021	0.035214	0.000109	0.007318
2012-01-11	0.000188	-0.002115	0.000113	0.007740

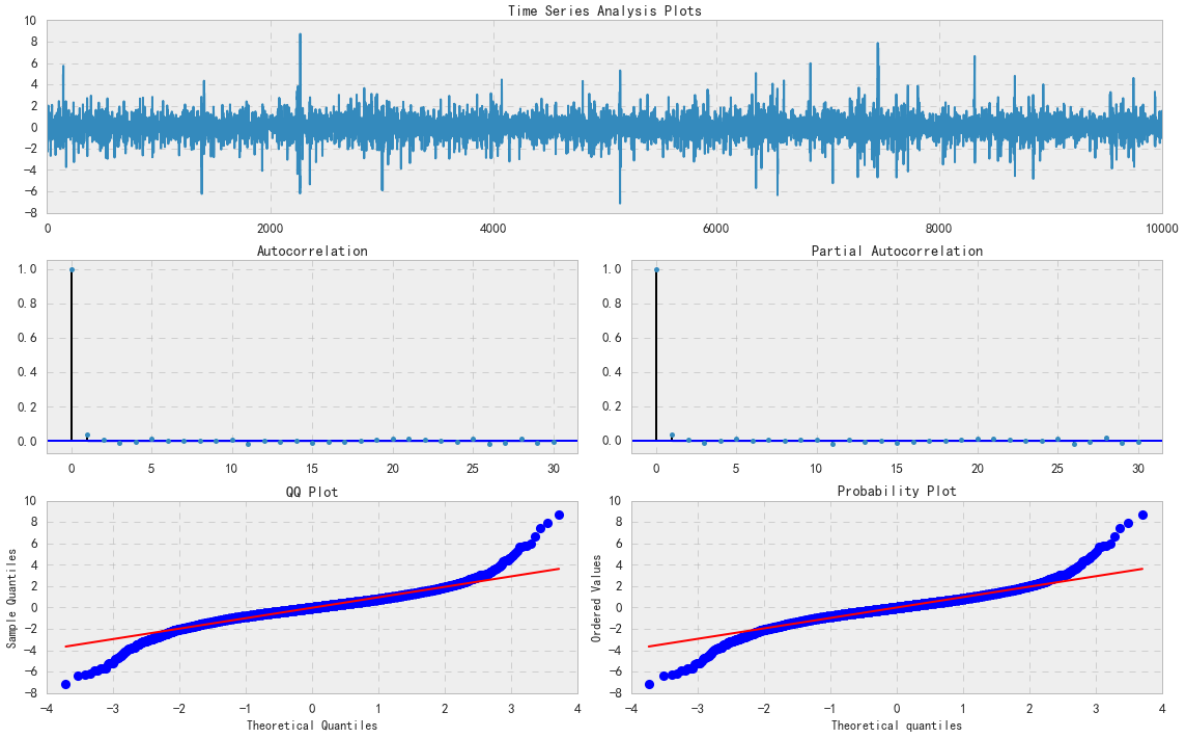
In [9]:

```
def tsplot(y, lags=None, figsize=(16, 10), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)
    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        #mpl.rcParams['font.family'] = 'Ubuntu Mono'
        layout = (3, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))
        qq_ax = plt.subplot2grid(layout, (2, 0))
        pp_ax = plt.subplot2grid(layout, (2, 1))

        y.plot(ax=ts_ax)
        ts_ax.set_title('Time Series Analysis Plots')
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax, alpha=0.5)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax, alpha=0.5)
        sm.qqplot(y, line='s', ax=qq_ax)
        qq_ax.set_title('QQ Plot')
        scs.probplot(y, sparams=(y.mean(), y.std()), plot=pp_ax)

        plt.tight_layout()
    return
```

我们先模拟一个GARCH(1,1)时序

In [10]:

```
# Simulating a GARCH(1, 1) process

np.random.seed(2)

a0 = 0.2
a1 = 0.5
b1 = 0.3

n = 10000
w = np.random.normal(size=n)
y = np.zeros_like(w)
sigsq = np.zeros_like(w)

for i in range(1, n):
    sigsq[i] = a0 + a1*(y[i-1]**2) + b1*sigsq[i-1]
    y[i] = w[i] * np.sqrt(sigsq[i])

_ = tsplot(y, lags=30)
```

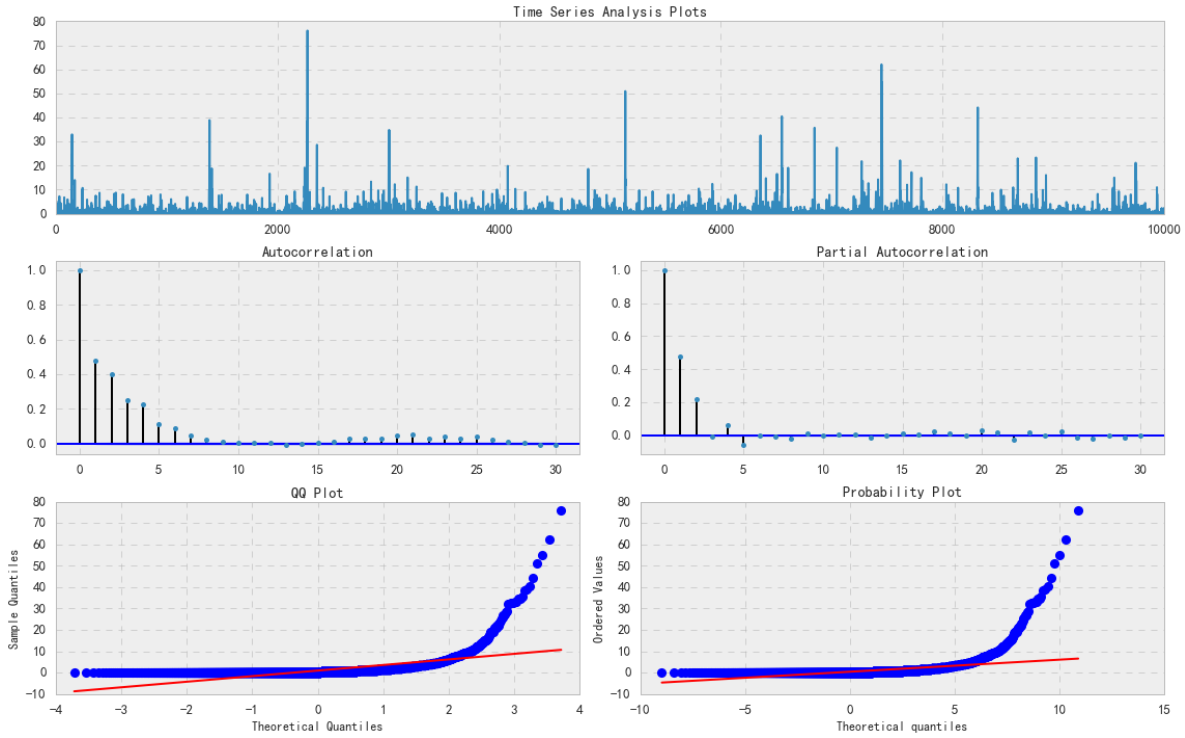


看上去很类似一个白噪声过程。

模拟时序的平方

In [11]:

```
tsplot(y*y, lags=30)
```



从ACF,PACF中显示出显著自相关性，我们需要AR项和MA项。

我们尝试是否可以通过模型拟合来得到模拟的参数。

In [12]:

```
# Fit a GARCH(1, 1) model to our simulated EPS series
# We use the arch_model function from the ARCH package

am = arch_model(y)
res = am.fit(update_freq=5)
print(res.summary())
```

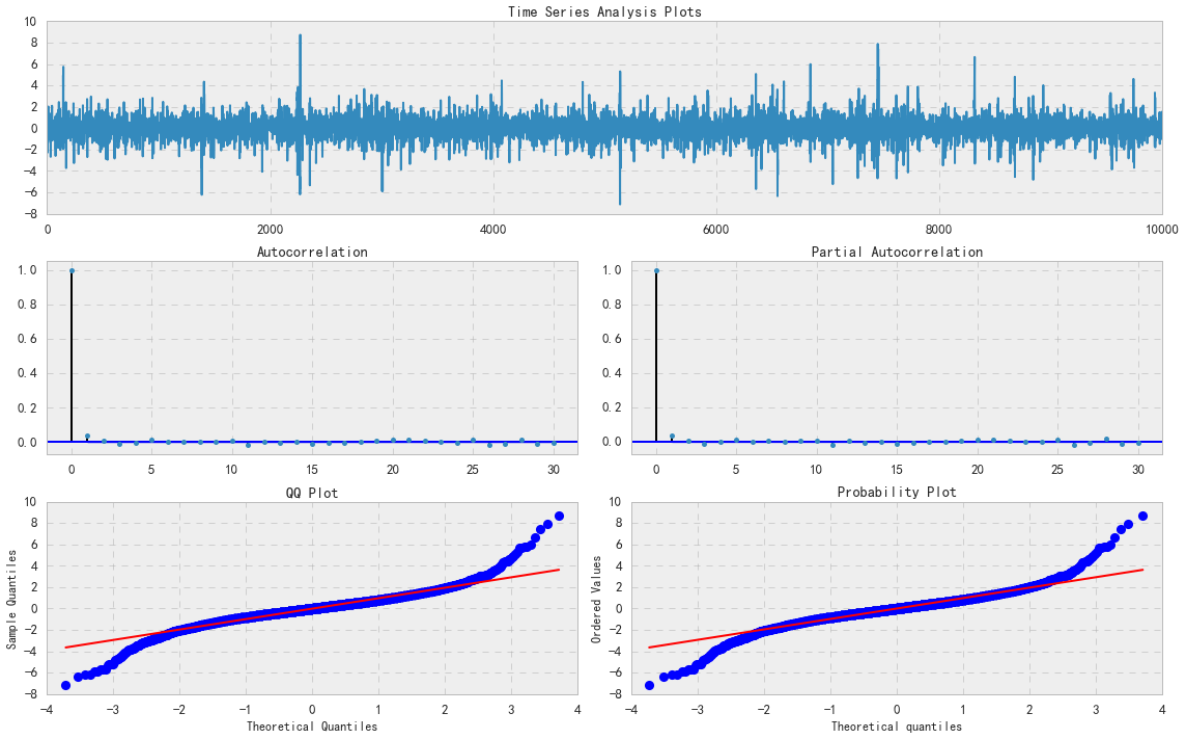
```
Iteration:      5,   Func. Count:     38,   Neg. LLF: 12311.793683614378
Iteration:     10,   Func. Count:     71,   Neg. LLF: 12238.592659128462
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 12237.303267318555
            Iterations: 13
            Function evaluations: 89
            Gradient evaluations: 13
                    Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                          y   R-squared:                      -0.000
Mean Model:                  Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                           GARCH   Log-Likelihood:               -12237.3
Distribution:                       Normal   AIC:                           24482.6
Method:                 Maximum Likelihood   BIC:                           24511.4
                                             No. Observations:                10000
Date:                     Sat, Jul 14 2018   Df Residuals:                     9996
Time:                             20:46:41   Df Model:                           4
                                  Mean Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -6.7225e-03  6.735e-03     -0.998      0.318 [-1.992e-02,6.478e-03]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.2021  1.043e-02     19.383  1.084e-83 [  0.182,  0.223]
alpha[1]       0.5162  2.016e-02     25.611 1.144e-144 [  0.477,  0.556]
beta[1]        0.2879  1.870e-02     15.395  1.781e-53 [  0.251,  0.325]
============================================================================

Covariance estimator: robust
```

我们较好地恢复了参数。

模拟数据GARCH残差plot

In [13]:

```
_ = tsplot(res.resid, lags=30)
```



以GARCH建模国内股票收益率

步骤如下：

1. 以ARIMA模型迭代得到最佳参数。
2. 以ARIMA模型所得到地具备最低AIC的参数来选择GARCH模型。
3. 使GARCH模型适配时序数据。
4. 检查模型残差和残差平方的自相关性。

In [14]:

```
def _get_best_model(TS):
    best_aic = np.inf
    best_order = None
    best_mdl = None

    pq_rng = range(5) # [0,1,2,3,4]
    d_rng = range(2) # [0,1]
    for i in pq_rng:
        for d in d_rng:
            for j in pq_rng:
                try:
                    tmp_mdl = smt.ARIMA(TS, order=(i,d,j)).fit(
                        method='mle', trend='nc'
                    )
                    tmp_aic = tmp_mdl.aic
                    if tmp_aic < best_aic:
                        best_aic = tmp_aic
                        best_order = (i, d, j)
                        best_mdl = tmp_mdl
                except: continue
    print('aic: {:6.5f} | order: {}'.format(best_aic, best_order))
    return best_aic, best_order, best_mdl

# Notice I've selected a specific time period to run this analysis
TS = indexs_logret['国内股票']
res_tup = _get_best_model(TS)
```

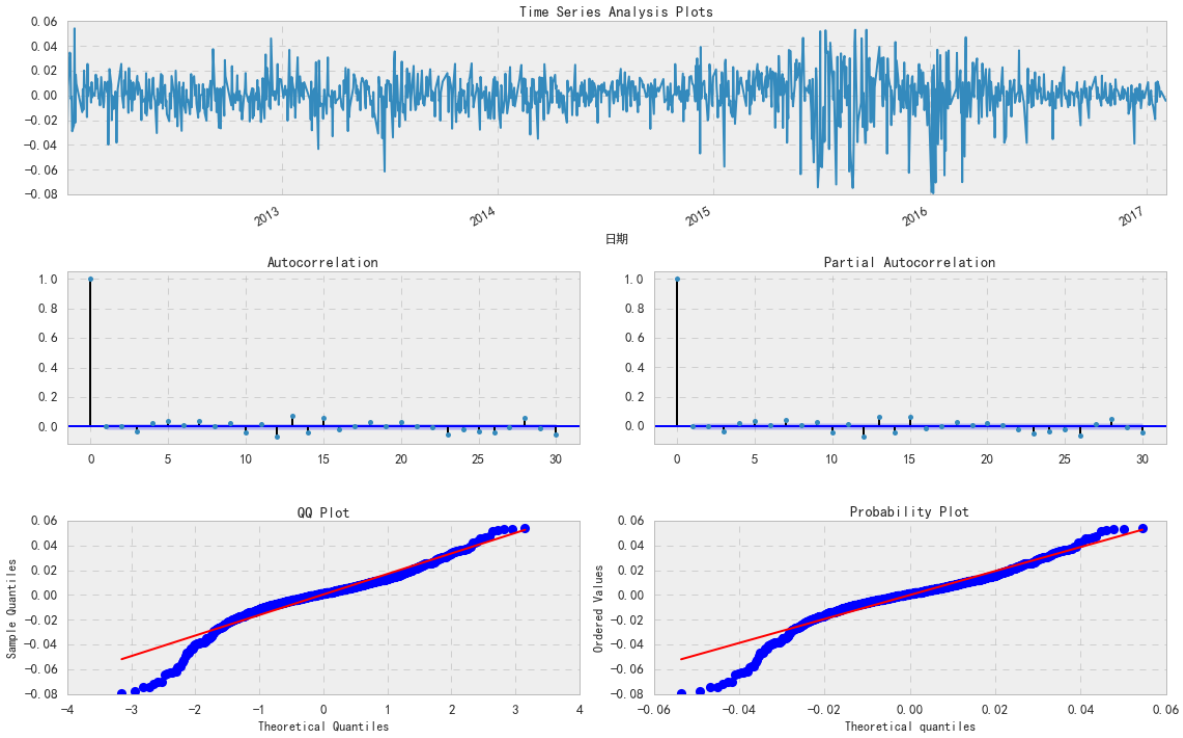
```
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)

aic: -6601.86081 | order: (3, 0, 2)
```

得到p,d,q = 3,0,2

残差Plot

In [15]:

```
_ = tsplot(res_tup[2].resid, lags=30)
```

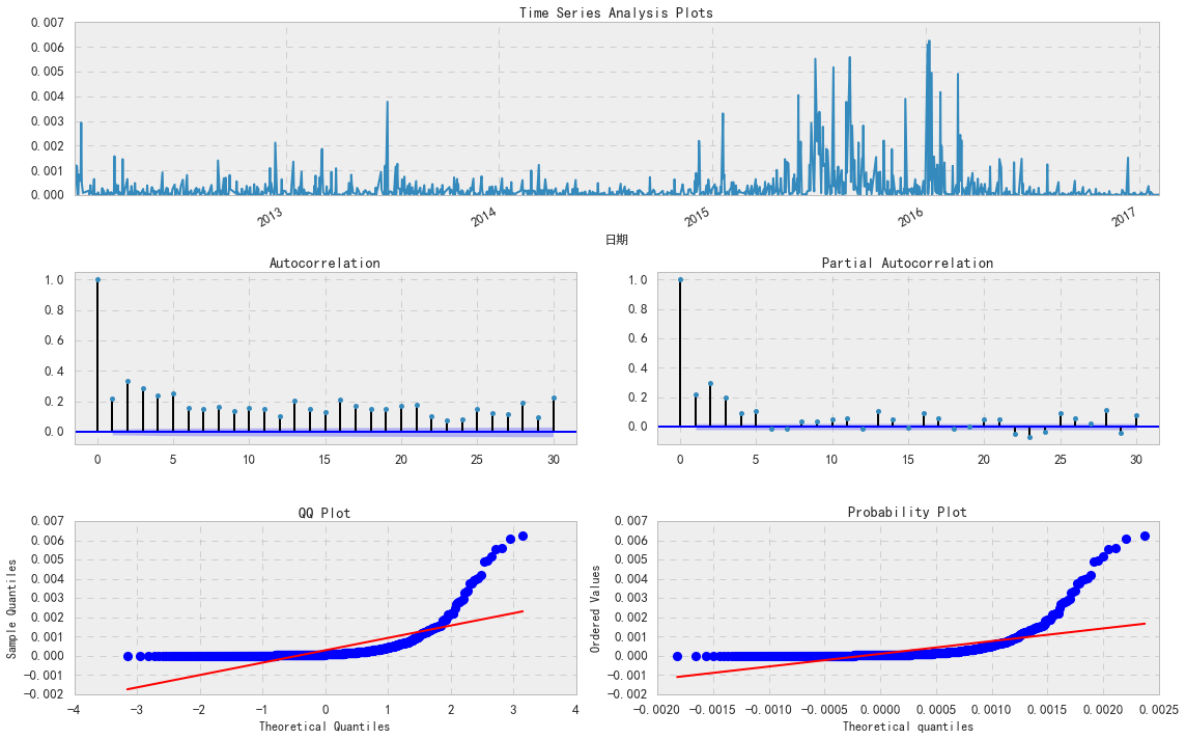


残差并非正态分布。

残差平方

In [16]:

```
_ = tsplot(res_tup[2].resid**2, lags=30)
```



残差平方显示较强的自相关性

拟合GARCH模型

In [17]:

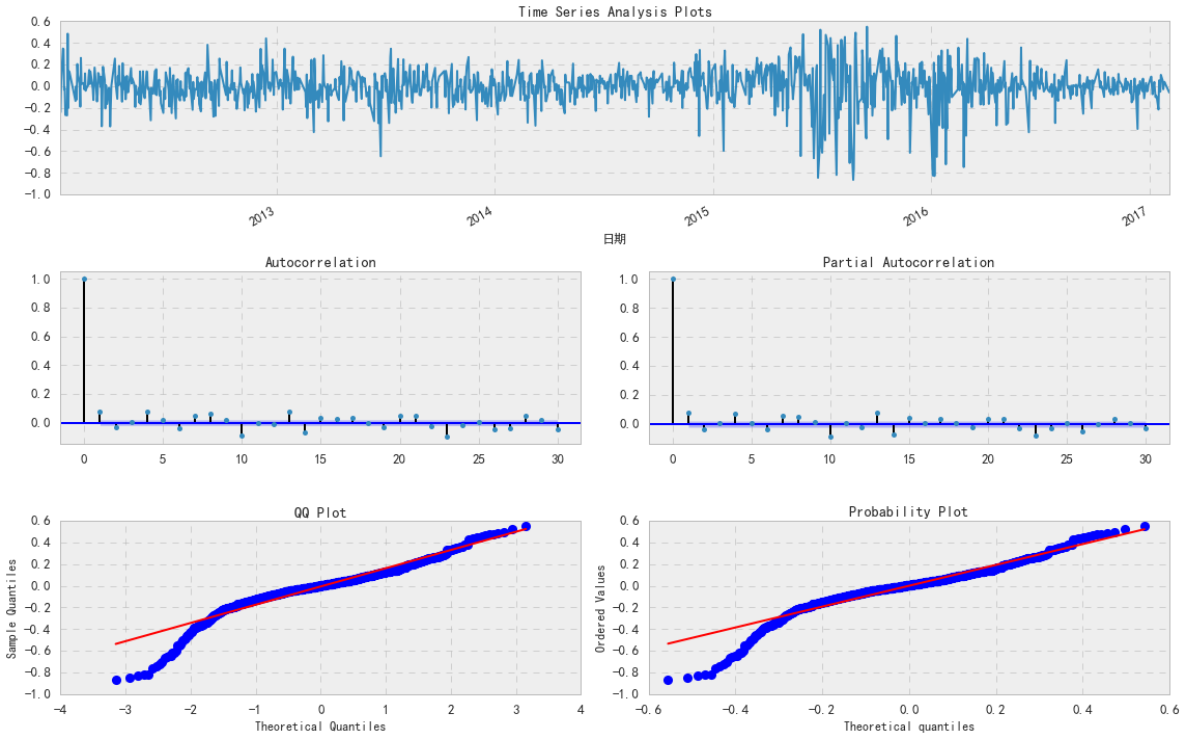
```
order = [3,0,2]
p_ = order[0]
o_ = order[1]
q_ = order[2]

# Using student T distribution usually provides better fit
am = arch_model(10*TS, p=p_, o=o_, q=q_, dist='StudentsT')
res = am.fit(update_freq=5, disp='off')
print(res.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                    国内股票   R-squared:                       -
0.001
Mean Model:             Constant Mean   Adj. R-squared:                  -0.001
Vol Model:                      GARCH   Log-Likelihood:                680.266
Distribution:      Standardized Student's t   AIC:                    -1344.53
Method:                 Maximum Likelihood   BIC:                    -1303.59
                                             No. Observations:             1234
Date:                  Sat, Jul 14 2018   Df Residuals:                   1226
Time:                          20:49:12   Df Model:                         8
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0114  3.373e-03      3.388  7.039e-04 [4.816e-03,1.804e-02]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega      5.2358e-04  2.969e-04      1.764  7.778e-02 [-5.827e-05,1.105e-03]
alpha[1]       0.0000  3.293e-02      0.000      1.000 [-6.454e-02,6.454e-02]
alpha[2]       0.0484  2.246e-02      2.155  3.116e-02  [4.382e-03,9.243e-02]
alpha[3]       0.0834  4.490e-02      1.858  6.315e-02   [-4.573e-03,    0.171]
beta[1]        0.0591  5.718e-02      1.034      0.301   [-5.293e-02,    0.171]
beta[2]        0.7923  5.724e-02     13.843  1.402e-43      [  0.680,    0.905]
                              Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu             4.9493      0.666      7.427  1.114e-13 [  3.643,   6.255]
============================================================================

Covariance estimator: robust
```

GARCH模型残差Plot

In [18]:

```
_ = tsplot(res.resid, lags=30)
```



残差平方自相关性依然显著，说明模型拟合不是很成功。

香港股票收益率GARCH拟合

步骤同上

In [20]:

```
TS = indexs_logret['香港股票']
res_tup = _get_best_model(10*TS)
```

```
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

```
aic: -1958.01617 | order: (2, 0, 2)
```

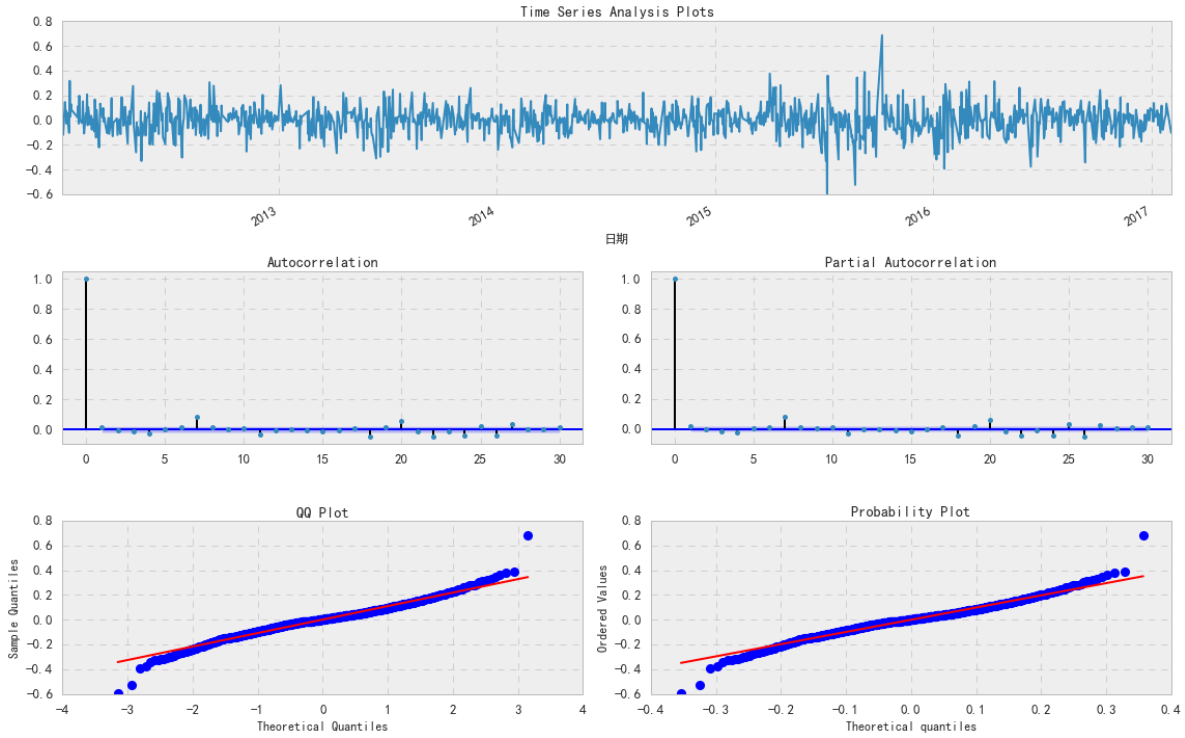
```
d:\Anaconda3\lib\site-packages\statsmodels\base\model.py:496: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

得到p,d,q=2,0,2

残差Plot

In [21]:

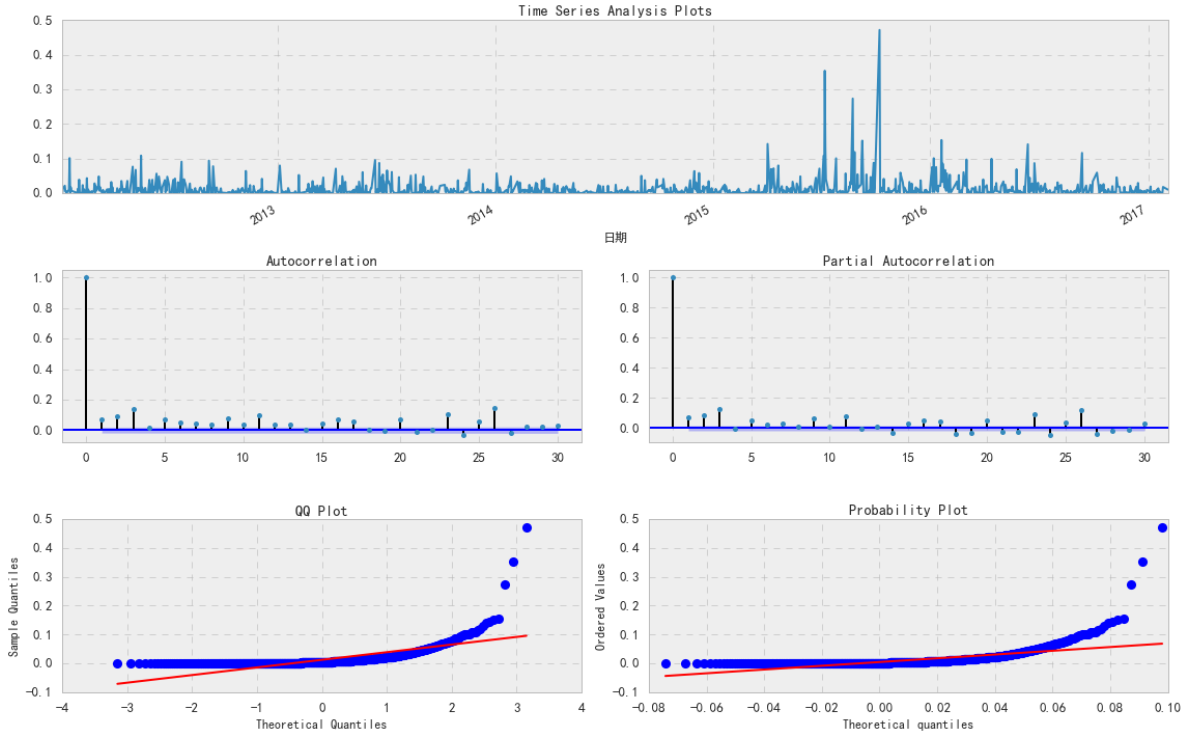
```
_ = tsplot(res_tup[2].resid, lags=30)
```



残差平方Plot

In [22]:

```
_ = tsplot(res_tup[2].resid**2, lags=30)
```



拟合GARCH模型

In [23]:

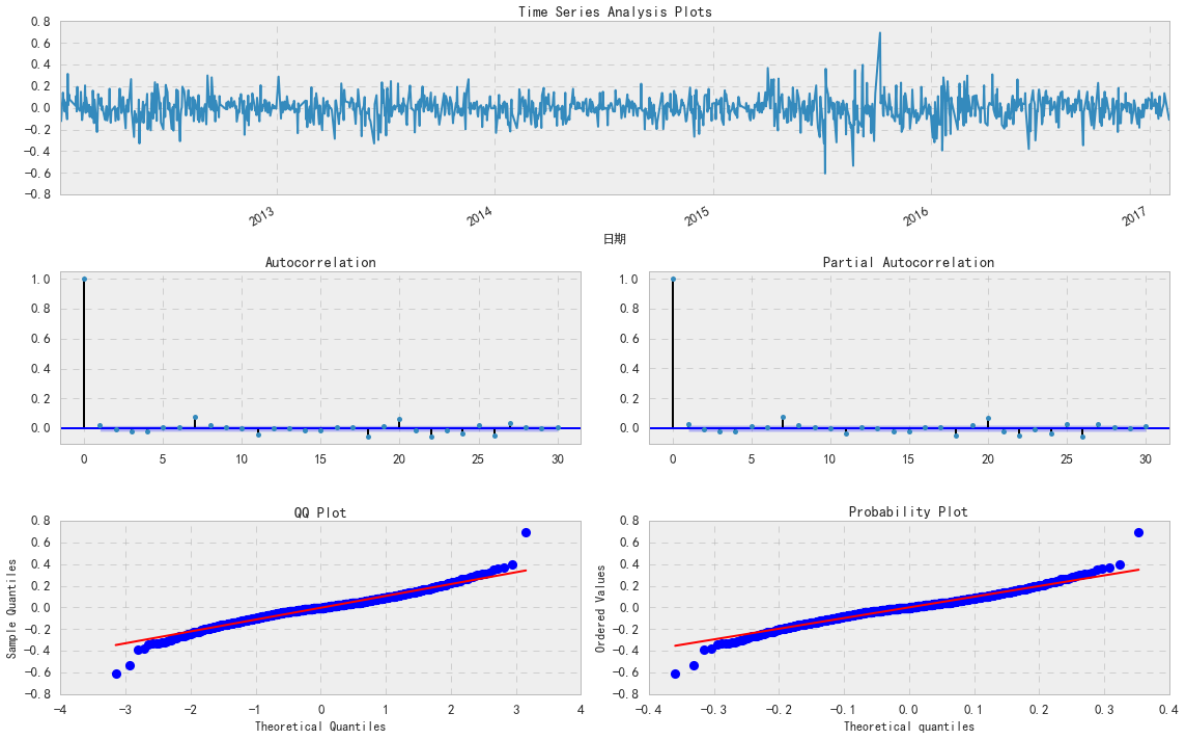
```
order = [2,0,2]
p_ = order[0]
o_ = order[1]
q_ = order[2]

# Using student T distribution usually provides better fit
am = arch_model(10*TS, p=p_, o=o_, q=q_, dist='StudentsT')
res = am.fit(update_freq=5, disp='off')
print(res.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                      香港股票   R-squared:                       -
0.001
Mean Model:                 Constant Mean   Adj. R-squared:                  -0.001
Vol Model:                          GARCH   Log-Likelihood:                1062.83
Distribution:      Standardized Student's t   AIC:                          -2111.66
Method:                  Maximum Likelihood   BIC:                          -2075.83
                                              No. Observations:                 1234
Date:                     Sat, Jul 14 2018   Df Residuals:                     1227
Time:                             20:55:56   Df Model:                            7
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
mu          4.8699e-03  2.613e-03      1.864  6.239e-02 [-2.520e-04,9.992e-03]
                           Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega       3.3282e-04  2.986e-04      1.115      0.265 [-2.524e-04,9.181e-04]
alpha[1]    2.7370e-10  6.923e-02  3.954e-09      1.000      [ -0.136,  0.136]
alpha[2]        0.0689  7.164e-02      0.961      0.337    [-7.157e-02,  0.209]
beta[1]         0.9077      0.487      1.864  6.236e-02    [-4.687e-02,  1.862]
beta[2]     3.9186e-09      0.497  7.883e-09      1.000      [ -0.974,  0.974]
                            Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu              4.9739      0.794      6.262  3.809e-10 [  3.417,  6.531]
============================================================================

Covariance estimator: robust
```

GARCH残差Plot

In [24]:

```
_ = tsplot(res.resid, lags=30)
```



拟合效果优于国内股票

总结

本文展示了采用Python语言为指数时序数据进行GARCH建模，并介绍了GARCH模型的基本概念。