模型-预测主题-连续型预测时间序列模型-ARIMA模型【hxy】

1. 模型名称

差分自回归移动平均模型（Autoregressive Integrated Moving Average Model，ARIMA）

2. 模型评价

2.1 优点

- 能够描述数据的**自回归性（autocorrelations）**

2.2 局限性

- 要求**平稳性（stationarity）**
  - 严平稳：表示的分布不随时间的改变而改变

    如白噪声（正态），无论怎么取，都是期望为0，方差为1
  - 弱平稳：期望与相关系数（依赖性）不变

    未来某时刻的t的值x_t要依赖于它的过去信息，所以需要依赖性
- 参数的选择会影响预测结果

3. 基本算法

3.1 定义

- **AR**：**Autoregessive 自回归**（用变量自身的历史时间数据对自身进行预测，必须满足平稳性要求）

$$p\text{阶自回归过程的公式定义：}y_t=\mu+\sum_{i=1}^{p}\gamma_i y_{t-i}+\epsilon_t$$

$$y_t\text{是当前值，}\mu\text{是常数项，}p\text{是阶数，}\gamma_i\text{是自相关系数，}\epsilon_t\text{是误差}$$

- **I**：**Integrated 差分的逆过程**

- **MA**：**Moving Average 移动平均**（自回归模型中的误差项的累加，有效消除预测中的随机波动）

$$q\text{阶自回归过程的公式定义：}y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-1}$$

- **ARMA**：**自回归移动平均模型**

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

- **ARIMA(p,d,q)**

$$p\text{是自回归项，}q\text{是移动平均项，}d\text{为时间序列称为平稳时所做的差分次数}$$

3.2 步骤

1. 画出**数据时序图**并检查有无异常观测
2. 如果必要的话，对数据进行变化（如**Box-Cox变换**）稳定方差
3. 如果数据非平稳，对数据进行**差分（difference）**直到数据平稳

$$\text{差分法：时间序列在}t\text{于}t-1\text{时刻的差值}$$

4. 检查**自相关图（ACF）**和**偏自相关图（PACF）**，判断数据是否有符合ARIMA(p,d,0)或ARIMA(0,d,q)模型的特征
   - **自相关函数（autocorrelation function，ACF）**

   $$ACF(k) = \rho_k = \frac{Cov(y_t, y_t - k)}{Var(y_t)}$$
   $$\rho_k\text{的取值范围为}[-1,1]$$

   - **偏自相关函数（partial autocorrelation function，PACF）**

     ACF还包含了中间其他变量的影响，而PACF是严格两个变量之间的相关性
   - 阶数确定（**截尾**指落在**95%的置信区间**内）

模型	**ACF**	**PACF**
AR(p)	衰减趋于零（几何型或振荡型）	p阶后截尾
MA(q)	q阶后截尾	衰减趋于零（几何型或振荡型）
ARMA(p,q)	q阶后衰减趋于零（几何型或振荡型）	p阶后衰减趋于零（几何形或振荡型）

5. 估计模型中未知参数的值（用**AIC**或**BIC**进行选择）

   通过**遍历**一定范围内的参数p和q，找到使得AIC或BIC**最小**的参数p和q
   - **赤池信息准则（Akaike Information Criterion，AIC）**

   $$AIC = 2k - 2ln(L)$$
   $$k\text{为模型参数个数，}n\text{为样本数量，}L\text{为似然函数}$$

   - **贝叶斯信息准则（Bayesian Information Criterion，BIC）**

$$BIC = kln(n) - 2ln(L)$$
$$k\text{为模型参数个数，}n\text{为样本数量，}L\text{为似然函数}$$

6. 通过画出**残差自相关图**来检查模型残差，如果残差类似白噪声，则进行模型预测
    - **类似白噪声**：符合平均值为0且方差为常数的正态分布
    - **QQ图**：线性即正态分布

4. 实例

4.1 数据介绍

[series1.csv]

Date	value
2006-06-01	0.21506609377014900
2006-07-01	1.142246186967090
2006-08-01	0.08077089285729770
2006-09-01	-0.7395189837372730
2006-10-01	0.5355162794384380
2006-11-01	-0.5647264651320740
2006-12-01	-1.1913935216543700
2007-01-01	-1.9961368164247100
2007-02-01	-1.8824096445368500
2007-03-01	-1.881361293860240
2007-04-01	-0.9766776697907430
2007-05-01	-1.9019923318711600
2007-06-01	-3.108610707028710
2007-07-01	-3.5268821422957000
2007-08-01	-2.7697700367119000
2007-09-01	-2.0338040672828000
2007-10-01	-3.180075966358300
2007-11-01	-3.3080815409072400
2007-12-01	-3.418501908245930
2008-01-01	-4.104801270845490

2008-02-01	-3.1744056756446300
2008-03-01	-1.425289135757780
2008-04-01	0.4401615904944530
2008-05-01	1.2679262510037600
2008-06-01	0.5441554324441390
2008-07-01	-0.48066970719551200
2008-08-01	-1.5799841374850200
2008-09-01	-0.1336163743808480
2008-10-01	1.7643398021524400
2008-11-01	-1.2648773342494300
2008-12-01	-3.1521978842334900
2009-01-01	-3.589928203018720
2009-02-01	-3.406228122833790
2009-03-01	-3.8263343229385600
2009-04-01	-2.7425294344933800
2009-05-01	-1.7887272597313700
2009-06-01	-2.4639028761016500
2009-07-01	-2.075657675328100
2009-08-01	-2.701547013802920
2009-09-01	-1.7025518581431700
2009-10-01	-0.7589319907020390
2009-11-01	-2.905804182464430
2009-12-01	-1.7551468831911900
2010-01-01	-1.9096679238947900
2010-02-01	-0.1291052757849870
2010-03-01	2.211945650117260
2010-04-01	1.569618562080830
2010-05-01	1.5087042686118500

2010-06-01	1.6998884025934600
2010-07-01	-1.7667376106183500
2010-08-01	-1.366051636217640
2010-09-01	0.7285490524822340
2010-10-01	2.2262658158365100
2010-11-01	1.6367586650124000
2010-12-01	0.043641470761556
2011-01-01	-2.3933886026916100
2011-02-01	-3.2353454025596100
2011-03-01	-2.101837771824050
2011-04-01	-0.8990375021239450
2011-05-01	-1.4936664574666800
2011-06-01	-3.1073167508272200
2011-07-01	-1.4205328314401400
2011-08-01	0.2758607214066030
2011-09-01	0.43735990925986600
2011-10-01	-0.2548263800847050
2011-11-01	-0.3458472155762090
2011-12-01	-0.6115030256444330
2012-01-01	-0.38337005302025600
2012-02-01	1.6954758499607500
2012-03-01	1.5613010884084200
2012-04-01	-0.17909432703339900
2012-05-01	-0.5810841195080400
2012-06-01	-2.3308344299399300
2012-07-01	-2.0057500970363400
2012-08-01	-0.5478467175272090
2012-09-01	0.7102722661376360

2012-10-01	1.5215664448259700
2012-11-01	1.323207097388040
2012-12-01	0.8370054084248230
2013-01-01	-0.10582093340683800
2013-02-01	-1.8597930129863500
2013-03-01	-1.9819951932431900
2013-04-01	-0.3690851457447420
2013-05-01	1.0213087568123700
2013-06-01	1.313503316487930
2013-07-01	1.138473914848280
2013-08-01	-0.5684114159918980
2013-09-01	-1.4298085649814000
2013-10-01	-1.8058074666928400
2013-11-01	-1.9511514403250400
2013-12-01	-1.4477675756531000
2014-01-01	-0.039660778396576300
2014-02-01	1.4280190595321100
2014-03-01	1.1451101579872400
2014-04-01	-1.6690214653425700
2014-05-01	-1.5040115349757500
2014-06-01	-2.448986495668510
2014-07-01	-2.8317230406994700
2014-08-01	-2.6938098802334200
2014-09-01	0.23414533840190500
2014-10-01	1.33963923299082
2014-11-01	1.4028876775484100
2014-12-01	1.7780474518983800
2015-01-01	1.6194943314372000

2015-02-01	0.4887985096857940
2015-03-01	2.208630445167100
2015-04-01	2.4556132237466800
2015-05-01	2.6470927240715700
2015-06-01	3.0162463456840600
2015-07-01	1.7039588266126300
2015-08-01	0.6037148295707650
2015-09-01	-1.2737209728501700
2015-10-01	-0.93284071310711
2015-11-01	0.08551545990148490
2015-12-01	1.20534410726747
2016-01-01	2.164110679279700
2016-02-01	0.9522611305039780
2016-03-01	0.3648520796360800
2016-04-01	-2.264868721883360
2016-05-01	-2.3816786375743700

4.2 实验目的

根据数据检验是否可以用ARIMA预测

4.3 代码实现

ARIMA.ipynb

```
%load_ext autoreload
%autoreload 2
%matplotlib inline
%config InlineBackend.figure_format='retina'

from __future__ import absolute_import, division, print_function

import sys
import os

import pandas as pd
import numpy as np
```

```python
# TSA from Statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt

# Display and Plotting
import matplotlib.pylab as plt
import seaborn as sns

pd.set_option('display.float_format', lambda x: '%.5f' % x) # pandas
np.set_printoptions(precision=5, suppress=True) # numpy

pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)

# seaborn plotting style
sns.set(style='ticks', context='poster')
```

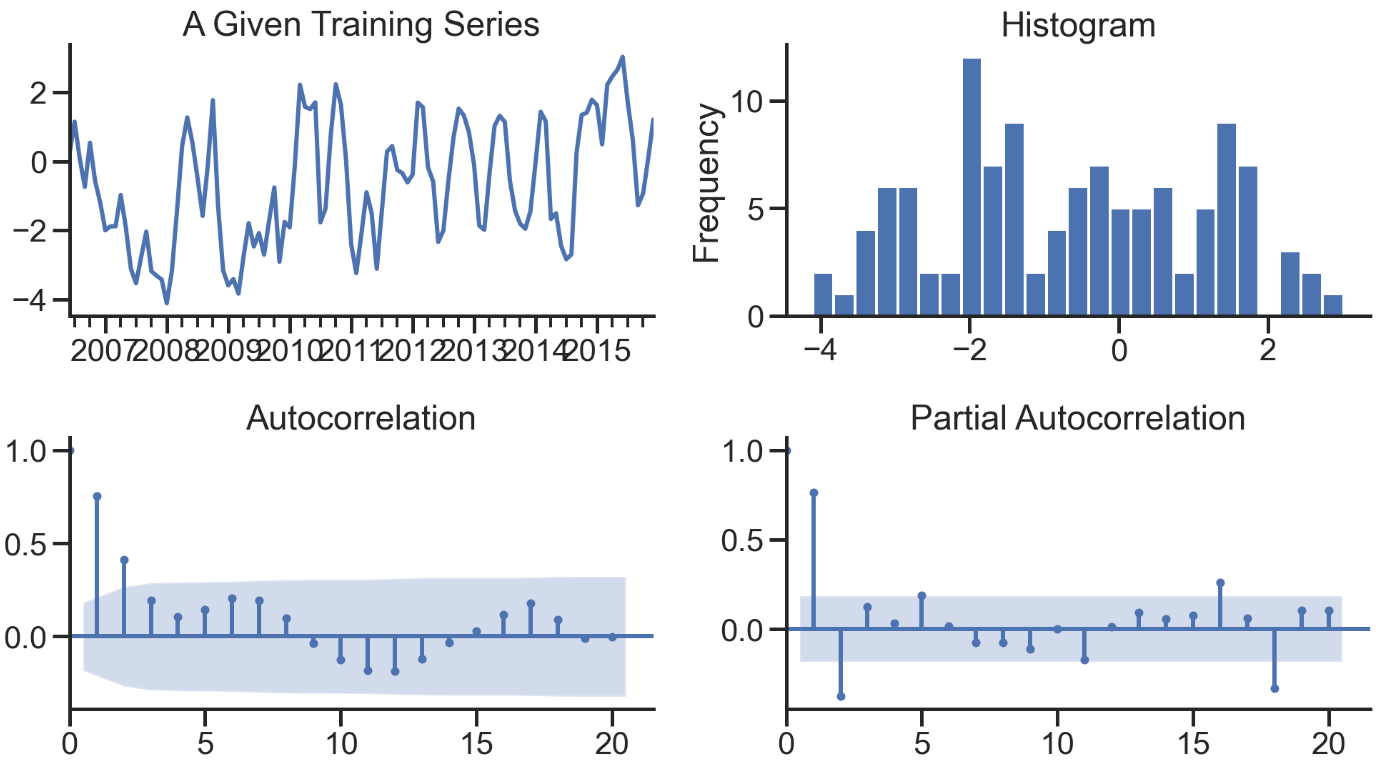
```python
# 读取数据
filename_ts = 'series1.csv'
ts_df = pd.read_csv(filename_ts, index_col=0, parse_dates=[0])
n_sample = ts_df.shape[0]
```

```python
# 划分测试集和训练集
n_train=int(0.95*n_sample)+1
n_forecast=n_sample-n_train
#ts_df
ts_train = ts_df.iloc[:n_train]['value']
ts_test = ts_df.iloc[n_train:]['value']
```

```python
# 画数据时序图（由于符合平稳性，未做差分），直方图，ACF和PACF
def tsplot(y, lags=None, title='', figsize=(14, 8)):

    fig = plt.figure(figsize=figsize)
    layout = (2, 2)
    ts_ax   = plt.subplot2grid(layout, (0, 0))
    hist_ax = plt.subplot2grid(layout, (0, 1))
    acf_ax  = plt.subplot2grid(layout, (1, 0))
    pacf_ax = plt.subplot2grid(layout, (1, 1))

    y.plot(ax=ts_ax)
    ts_ax.set_title(title)
    y.plot(ax=hist_ax, kind='hist', bins=25)
    hist_ax.set_title('Histogram')
    smt.graphics.plot_acf(y, lags=lags, ax=acf_ax)
    smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax)
    [ax.set_xlim(0) for ax in [acf_ax, pacf_ax]]
```

```
    sns.despine()
    fig.tight_layout()
    return ts_ax, acf_ax, pacf_ax
```

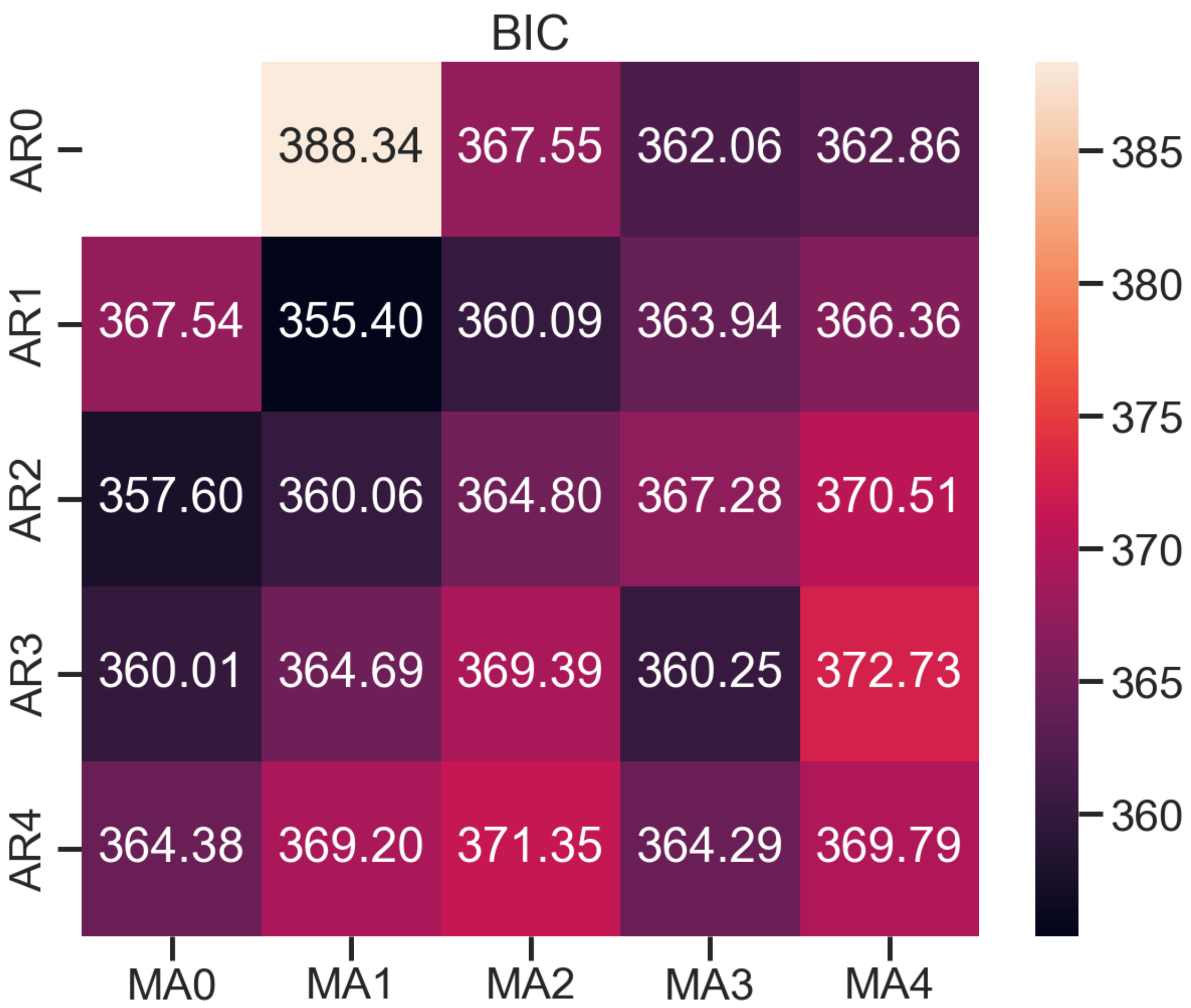
```
tsplot(ts_train, title='A Given Training Series', lags=20);
```




```
# 遍历用BIC寻找最优参数
import itertools

p_min = 0
d_min = 0
q_min = 0
p_max = 4
d_max = 0
q_max = 4

# Initialize a DataFrame to store the results
results_bic = pd.DataFrame(index=['AR{}'.format(i) for i in range(p_min,p_max+1)],
                           columns=['MA{}'.format(i) for i in range(q_min,q_max+1)])

for p,d,q in itertools.product(range(p_min,p_max+1),
                               range(d_min,d_max+1),
                               range(q_min,q_max+1)):
    if p==0 and d==0 and q==0:
        results_bic.loc['AR{}'.format(p), 'MA{}'.format(q)] = np.nan
        continue

    try:
```

```
        model = sm.tsa.SARIMAX(ts_train, order=(p, d, q),
                               #enforce_stationarity=False,
                               #enforce_invertibility=False,
                              )
        results = model.fit()
        results_bic.loc['AR{}'.format(p), 'MA{}'.format(q)] = results.bic
    except:
        continue
results_bic = results_bic[results_bic.columns].astype(float)
```

```
fig, ax = plt.subplots(figsize=(10, 8))
ax = sns.heatmap(results_bic,
                 mask=results_bic.isnull(),
                 ax=ax,
                 annot=True,
                 fmt='.2f',
                 );
ax.set_title('BIC');
```

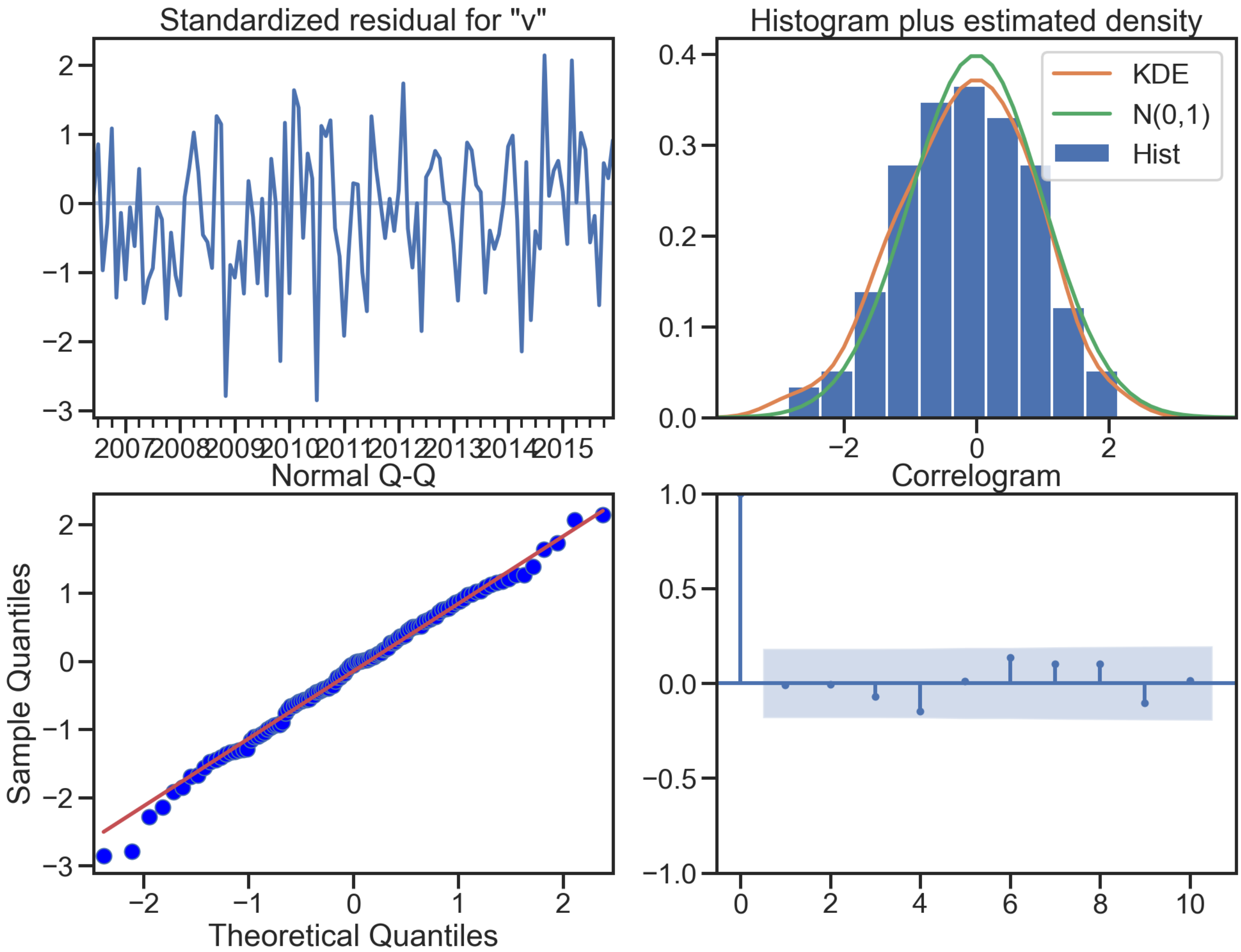


```python
# 用AIC和BIC算出最优参数

train_results = sm.tsa.arma_order_select_ic(ts_train, ic=['aic', 'bic'], trend='nc',
max_ar=4, max_ma=4)

print('AIC', train_results.aic_min_order)
print('BIC', train_results.bic_min_order)
```

```
AIC (4, 2)
BIC (1, 1)
```

```python
# 用合适的参数训练模型order=(p,d,q)
# 根据AIC可选order=(4,0,2)
# 根据BIC可选order=(1,0,1)
arima200 = sm.tsa.SARIMAX(ts_train, order=(1,0,1))
model_results = arima200.fit()
```

```python
#残差分析 正态分布 QQ图线性
model_results.plot_diagnostics(figsize=(16, 12));
```



由图可知：残差符合类似白噪声，可用于做预测

5. 参考资料

1. [唐宇迪Python数据分析机器学习-时间序列分析]
2. [机器学习（五）——时间序列ARIMA模型]
3. 高老师第三次培训PPT：P100-P102