
高级量化交易技术

闫涛
科技有限公司
北京
{yt7589}@qq.com

第 1 章 时间序列基本特性

Abstract

在本章中我们将讨论时间序列的基本特性，包括自相关性和平稳性。

1 时间序列基本特性

时间序列的自相关性是指时间序列过去与未来存在某种关系，是我们时间序列预测的基础。主要用自协方差函数 (Autocovariance Function, AF)、自相关系数函数 (Autocorrelation Coefficient Function, ACF) 和偏自相关系数函数 (Partial Autocorrelation Coefficient Function, PACF) 来描述。

1.1 随机变量统计量

随机变量 X 其取值为 x 的均值定义为：

$$E(x) = \mu \quad (1)$$

方差定义为：

$$\sigma^2(x) = E\left[(x - \mu)^2\right] \quad (2)$$

其中 $\sigma(x)$ 为标准差。对于两个随机变量 x 和 y ，其协方差可以定义为：

$$\sigma(x, y) = E\left[(x - \mu_x)(y - \mu_y)\right] \quad (3)$$

在实际应用中，我们不可能知道真实的均值，只能使用统计量，因此协方差可以定义为：

$$Cov(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (4)$$

随机变量 x 和 y 的相关系数定义为：

$$\rho(x, y) = \frac{E[(x - \mu_x)(y - \mu_y)]}{\sigma_x \sigma_y} = \frac{\sigma(x, y)}{\sigma_x \sigma_y} \quad (5)$$

采用统计量的表示方法为：

$$Cor(x, y) = \frac{Cov(x, y)}{std(x) \times std(y)} \quad (6)$$

以上我们讨论的都是不同随机变量之间的关系，对于时间序列来说，我们可以把从不同时间点开始的子时间序列，视为不同的随机变量，那么我们就可以定义自协方差、自相关系数函数和偏自相关系数函数了。

1.2 时序序列平稳性

时序信号 x_t 的均值定义为：

$$E(x_t) = \mu(t) \quad (7)$$

时间序列的均值与所考虑的时间点有关。我们可以把时间序列上每个时间点都视为一个独立的时间变量，但是对于时间序列而言，每个时间点的随机变量只有一个观测值，怎么求出均值呢？在实际应用中，我们会将时间序列中的趋势信号（上涨或下跌）、季节性信号等从时间序列中去除掉，对于剩下的残差序列，我们可以视为其各个时间点上的随机变量的均值是不变的，于是就可以使用下面的公式来计算均值：

$$\bar{x} = \frac{1}{N} \sum_{t=1}^N x_t \quad (8)$$

由此我们引入平稳时间序列的概念，对于平稳时间序列，其各个时间点上对应的随机变量的均值相等。时间序列的方差可以定义为：

$$\sigma^2(t) = E[(x_t - \mu)^2] \quad (9)$$

根据上面平稳时间序列的定义，各个时间点对应的随机变量的均值不变，则式9可以化简为：

$$\sigma^2(t) = E[(x_t - \mu)^2] \quad (10)$$

我们同时规定，平稳时间序列各个时间点对应的随机变量的方差也不变，则10可进一步化简为：

$$Var(x_t) = \frac{1}{N-1} \sum_{t=1}^N N(x_t - \bar{x})^2 \quad (11)$$

1.3 自协方差

在讨论自协方差之前，我们首先要定义二阶平稳性。根据上一节定义，平稳时间序列是指各个时间点对应的随机变量的均值和方差相同。二阶平稳性是指在这一基础上，不同时间点对应的随机变量的相关系数只与时间相隔（lag）相关。注意：以下我们讨论的各种性质，均以此为前提。对于 lag=k 的自协方差定义为：

$$C_k = E[(x_t - \mu)(x_{t+k} - \mu)] \quad (12)$$

1.4 自相关系数函数 ACF

由于自协方差的大小与随机变量的大小有关，无法准确衡量其间的关系，因此我们引入自相关系数函数 ACF：

$$\rho_k = \frac{C_k}{\sigma^2} \quad (13)$$

由定义可知：

$$\begin{aligned} \rho_0 &= \frac{C_0}{\sigma^2} = \frac{E[(x_t - \mu)(x_t - \mu)]}{\sigma^2} \\ &= \frac{E[(x_t - \mu)^2]}{\sigma^2} = \frac{\sigma^2}{\sigma^2} = 1 \end{aligned} \quad (14)$$

在实际应用中，我们都是处理的离散数据点，则自协方差可以定义为：

$$c_k = \frac{1}{N} \sum_{t=1}^N (x_t - \bar{x})(x_{t+k} - \bar{x}) \quad (15)$$

自相关系数函数 ACF 可以定义为：

$$r_k = \frac{c_k}{c_0} \quad (16)$$

1.5 自相关性举例

下面我们以上证综指时间序列为例，来自相关系数函数 ACF 和偏自相关系数函数 PACF 的求法和作图，程序代码如下所示：

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import matplotlib.dates as mdates
4 from matplotlib.font_manager import FontProperties
5 from statsmodels.tsa import stattools
6 from statsmodels.graphics import tsaplots
7
8 class Chp023(object):
9     def __init__(self):
10         self.name = 'Chp022'
11         # 数据文件格式：编号 日期 星期几 开盘价 最高价
12         # 最低价 收益价 收益
13         self.data_file = 'data/pqb/chp023_001.txt'
14
15     def startup(self):
16         print('第23章：时间序列基本性质')
17         data = pd.read_csv(self.data_file, sep='\t', index_col='Trddt')
18         sh_index = data[Indexcd==1]
19         sh_index.index = pd.to_datetime(sh_index.index)
20         sh_return = sh_index.Retindex
21         print('时间序列长为：N={0}'.format(len(sh_return)))
22         acfs = stattools.acf(sh_return)
23         print(acfs)
24         pacfs = stattools.pacf(sh_return)
25         print(pacfs)
26         tsaplots.plot_acf(sh_return, use_vlines=True, lags=30)
27         plt.show()
28         tsaplots.plot_pacf(sh_return, use_vlines=True, lags=30)
29         plt.show()
```

Listing 1: 时间序列基本性质

数据文件格式为：

Figure 1: 数据文件格式

	Indexcd	Trddt	Daywk	Opnindex	Hiindex	Loindex	Clsindex	Retindex
1	2014/1/2	4	2112.126	2113.11	2101.016	2109.387		-0.003115
1	2014/1/3	5	2101.542	2102.167	2075.899	2083.136		-0.012445
1	2014/1/6	1	2078.684	2078.684	2034.006	2045.709		-0.017967
1	2014/1/7	2	2034.224	2052.279	2029.246	2047.317		0.000786
1	2014/1/8	3	2047.256	2062.952	2037.11	2044.34		-0.001454
1	2014/1/9	4	2041.773	2057.196	2026.446	2027.622		-0.008178
1	2014/1/10	5	2023.535	2029.297	2008.007	2013.298		-0.007064
1	2014/1/13	1	2014.978	2027.181	2000.404	2009.564		-0.001855

其运行结果为：

Figure 2: 运行结果

```
量化投资以python为工具
第23章：时间序列基本性质
时间序列长为：N=311
[ 1.  0.03527714 -0.01178861 -0.02953388  0.16043181 -0.0506902
-0.00557277  0.02556123  0.01763209  0.01170585  0.05137502 -0.03961812
 0.00219185  0.06976089  0.07020637 -0.0165844  0.09777829  0.10084446
 0.04706095 -0.05291647  0.08159786 -0.04505366  0.04894213 -0.11532665
 0.04273513 -0.04828588  0.04383656  0.06780466 -0.0642314  -0.05893641
-0.107162  0.05179026 -0.04171157  0.08340151  0.05368795  0.04874735
-0.04272709  0.03709712 -0.01313166 -0.04551243 -0.07680743]
[ 1.  0.03539094 -0.01313388 -0.02897258  0.16483494 -0.06656327
 0.00233848  0.03690183 -0.01674841  0.03307673  0.05240115 -0.05902877
 0.01423925  0.07184167  0.04734467  0.00131723  0.10809391  0.07778389
 0.03637376 -0.04041105  0.06539274 -0.07473672  0.05527639 -0.1271187
 0.02235463 -0.03279396  0.00769879  0.10885562 -0.10892202 -0.04912167
-0.16407526  0.0362806 -0.04578521  0.106876  0.05526165  0.04072968
-0.03517882  0.04134873 -0.02148459 -0.03297278 -0.08411357]
```

在图2中，第一个数据为自相关系数函数 ACF 各期的值，而第二个数组为偏自相关系数函数 PACF 各期的值。判断时间序列是否具有自相关性，可以看除 ACF 和 PACF 中，除第一个元素外，有没有显著超过阈值的元素，阈值定义为：

$$threshold = \frac{1.96}{\sqrt{N}} = \frac{1.96}{\sqrt{311}} = 0.11 \quad (17)$$

式17中的 N 为时间序列样本数，在本例中，共有 311 条记录，故 $N=311$ ，所以其阈值为 0.11 左右。由于 $acf[4] = 0.16 > 0.11$ 所以可以推断其具有自相关性，同时 $pacf[4] = 0.165 > 0.11$ 也可以推断其具有自相关性。我们还可以通过图形的方式形象的表示出来，自相关系数函数图如所示：

Figure 3: 自相关系数函数图

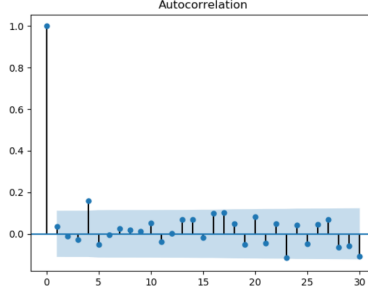
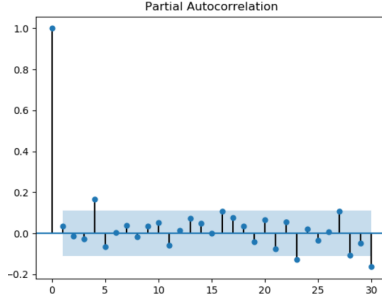


图3中蓝色区域的界限为 $[-\frac{1.96}{\sqrt{N}}, \frac{1.96}{\sqrt{N}}] = [-0.11, 0.11]$ ，除第 1 项外，其他项如果超出蓝色区域则说明此时间序列具有自相关性。偏自相关系数函数图为：

Figure 4: 偏自相关系数函数图



1.6 白噪声和随机游走

1.6.1 残差序列定义

我们要对任意时间序列 y_t 进行建模，我们的模型为 \hat{y}_t ，残差序列 x_t 可以定义为： $x_t = y_t - \hat{y}_t$ ，我们的任务就是使残差时间序列中每一个时间点对应的随机变量互相独立，没有自相关性，即满足独立同分布（Independent and Identical Distribution, I.I.D）条件。如果各个随机变量 $x_t \sim \mathbb{N}(0, \sigma^2)$ ，则称其为高斯白噪声。

1.6.2 差分运算符

为了后续讨论问题方便，我们首先定义 BSO 运算符：

$$Bx_t = x_{t-1} \quad B^n x_t = x_{t-n} \quad (18)$$

我们定义差分运算符为：

$$\begin{aligned} \nabla x_t &= x_t - x_{t-1} = (1 - B)x_t \\ \nabla^n x_t &= (x_t - x_{t-n})^n = (1 - B)^n x_t \end{aligned} \quad (19)$$

1.6.3 白噪声定义

对于时间序列 $\{w_t, t = 1, 2, 3, \dots, N\}$, 满足 $\forall t \quad w_t \sim \mathcal{N}(0, \sigma^2)$ 且 $\forall i \neq j \quad Cor(w_i, w_j) = 0$, 则其为白噪声时间序列。

下面我们来看白噪声的二阶属性:

$$\begin{aligned} \mu &= E(w_t) = 0 \\ \gamma_k &= Cor(w_t, w_{t+k}) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases} \end{aligned} \quad (20)$$

1.6.4 随机游走

随机游走 (Random Walk) 时间序列是指 x_t 可以定义为: $x_t = x_{t-1} + w_t$, 其中 w_t 为白噪声时间序列。随机游走时间序列可以表示为:

$$\begin{aligned} x_t &= x_{t-1} + w_t = Bx_t + w_t \\ x_t &= x_{t-1} + w_t = x_{t-2} + w_{t-1} + w_t \\ &\dots\dots \\ x_t &= w_1 + w_2 + \dots + w_{t-1} + w_t \end{aligned} \quad (21)$$

所以随机游走时间序列可以看作是多个白噪声时间序列的叠加。下面我们来看随机游走时间序列的均值和协方差:

$$\begin{aligned} \mu &= 0 \\ \gamma_k(t) &= Cov(x_t, x_{t+k}) = t\sigma^2 \end{aligned} \quad (22)$$

我们再来看随机游走序列的自相关系数函数 ACF:

$$\begin{aligned} \rho_k(t) &= \frac{Cov(x_t, x_{t+k})}{\sqrt{Var(x_t) \cdot Var(x_{t+k})}} \\ &= \frac{t\sigma^2}{\sqrt{t\sigma^2(t+k)\sigma^2}} = \frac{1}{\sqrt{1 + \frac{k}{t}}} \end{aligned} \quad (23)$$

在通常情况下, t 比 k 要大得多, 因此 ρ_k 会比较接近于 1。

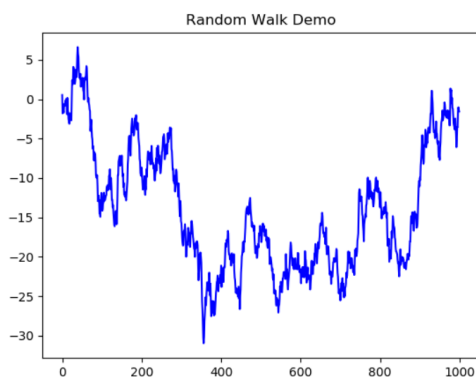
下面我们来模拟一个随机游走时间序列信号, 程序如下所示:

```
1 def random_wale_demo(self):
2     '''
3     随机游走时间序列建模示例
4     '''
5     w = np.random.standard_normal(size=1000)
6     x = w
7     for t in range(1, len(w)):
8         x[t] = x[t-1] + w[t]
9     plt.plot(x, c='b')
10    plt.title('Random Walk Demo')
11    plt.show()
12    acfs = stattools.acf(x)
13    print(acfs)
14    tsaplots.plot_acf(x, use_vlines=True, lags=30)
15    plt.show()
16    # 拟合随机游走信号
17    r = []
18    for t in range(1, len(x)):
19        r.append(x[t] - x[t-1])
20    rd = np.array(r)
21    plt.plot(rd, c='r')
22    plt.title('Residue Signal')
23    plt.show()
24    rd_acfs = stattools.acf(rd)
25    print(rd_acfs)
26    tsaplots.plot_acf(rd, use_vlines=True, lags=30)
```

Listing 2: 随机游走过程模拟

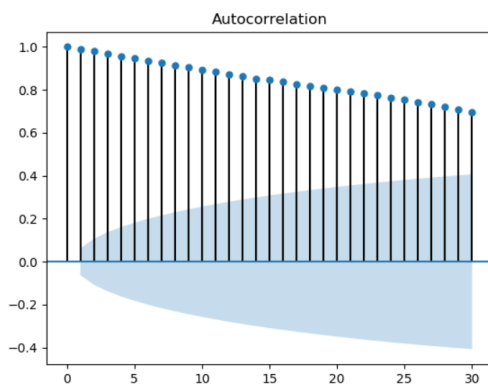
我们首先通过 $x_t = x_{t-1} + w_t$ 生成一个随机游走信号，该信号图形如下所示：

Figure 5: 随机游走时间序列信号



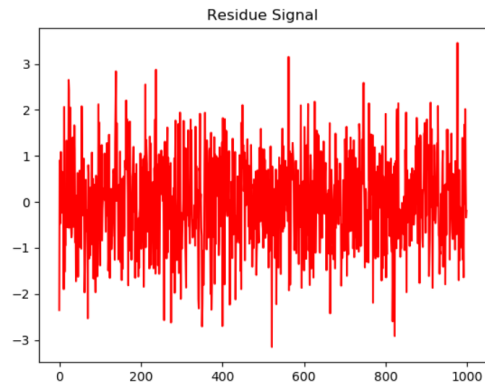
由图5可以看出，其非常像是一个股票收盘价的走势图，这也是为什么有些人说股票走势是随机游走过程了。接着我们求出该时间序列的自相关系数函数 ACF 及其自相关图，如下所示：

Figure 6: 随机游走时间序列信号



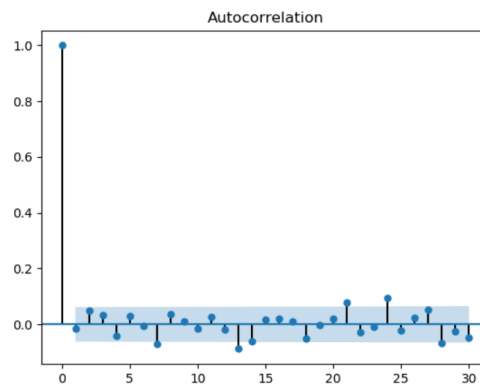
由图可以看出，其具有极强的自相关性，所有 ACF 值均位于蓝色置信区间之外。我们知道 $x_t - x_{t-1} = w_t$ ，而 w_t 是白噪声时间序列信号，这实际上模拟了实际应用过程，我们把 x_t 视为实际的金融信号，而 x_{t-1} 为我们建模的信号，将两个信号相减，得到残差信号，如果残差信号是白噪声信号，就可以认为我们建模是合理的。下面来看我们得到的残差信号：

Figure 7: 残差时间序列信号



计算并绘制 ACF 如下所示：

Figure 8: 残差自相关系数函数



程序的运行结果如下所示：

Figure 9: 程序运行结果

```
量化投资以python为工具
第23章：时间序列基本性质
[1. 0.98914742 0.97903129 0.96783309 0.95632931 0.94596796
 0.93489265 0.92427096 0.91497217 0.90461824 0.89396843 0.88323481
 0.87235068 0.86216894 0.85285975 0.84479971 0.83634853 0.82763235
 0.81879369 0.81058584 0.80226346 0.79341307 0.78342443 0.77367788
 0.76487486 0.75417161 0.74313034 0.73174875 0.71939583 0.7078342
 0.6965237 0.6861672 0.6760541 0.66486422 0.65380632 0.64342133
 0.63339216 0.62341977 0.6128232 0.60296489 0.59324119]
[ 1. 0.0000000e+00 -1.48521639e-02 5.06985700e-02 3.21625338e-02
 -4.24342295e-02 2.86756161e-02 -4.41871354e-03 -7.18986605e-02
 3.66200199e-02 1.21850369e-02 -1.59723171e-02 2.68915024e-02
 -1.94434230e-02 -8.78575550e-02 -5.94851957e-02 1.62981893e-02
 1.93119858e-02 8.95837565e-03 -4.99698205e-02 -3.38587649e-03
 2.00536805e-02 7.92334588e-02 -2.79786314e-02 -8.84736489e-03
 9.60908594e-02 -2.21958482e-02 2.21536791e-02 5.16706070e-02
 -6.80076985e-02 -2.63520397e-02 -4.69741376e-02 7.80032912e-04
 7.86023537e-02 -2.33883445e-02 -5.47155569e-02 -2.65466308e-02
 -3.87128530e-03 2.09699285e-02 -5.18812130e-02 -2.84822052e-02
 1.58564831e-02]
```

下面我们以上证综指收益率为例，来看随机游走模型是否可以很好的拟合这个时间序列，程序如下所示：

```
1 def random_walk_fit(self):
2     data = pd.read_csv(self.data_file, sep='\t', index_col='Trddt')
3     sh_index = data[data.Indexcd==1]
4     sh_index.index = pd.to_datetime(sh_index.index)
```

```

5 sh_return = sh_index.Retindex
6 print('时间序列长为: N={0}'.format(len(sh_return)))
7 r = []
8 for t in range(1, len(sh_return)):
9     r.append(sh_return[t] - sh_return[t-1])
10 rd = np.array(r)
11 plt.plot(rd, c='b')
12 plt.title('Random Walk fit SHIndex Return')
13 plt.show()
14 rd_acfs = stattools.acf(rd)
15 print(rd_acfs)
16 tsaplots.plot_acf(rd, use_vlines=True, lags=30)
17 plt.show()

```

Listing 3: 随机游走拟合上证综指收益率

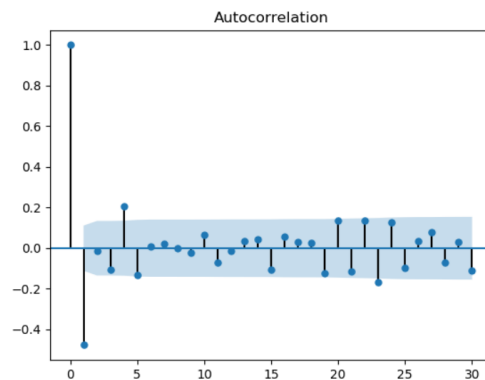
其残差图像为:

Figure 10: 残差图形



自相关系数函数 ACF 图形:

Figure 11: 自相关系数函数 ACF



由图11所示, 在 1、4 时间点, 明显超出置信范围, 因此随机游走过程不能很好的拟合上证综指收益率时间序列信号。程序的运行结果如下所示:

Figure 12: 程序运行结果

```

量化投资以python为工具
第23章：时间序列基本性质
时间序列长为：N=311
[ 1.00000000e+00 -4.76495633e-01 -1.54901215e-02 -1.06305387e-01
 2.07607804e-01 -1.33423618e-01 7.48593453e-03 2.06096879e-02
-6.05232517e-04 -2.38979190e-02 6.74201467e-02 -6.91058174e-02
-1.32615416e-02 3.58256731e-02 4.58501980e-02 -1.05971579e-01
 5.84057765e-02 2.86644902e-02 2.44271741e-02 -1.21337912e-01
 1.34541909e-01 -1.13442293e-01 1.34590834e-01 -1.67581384e-01
 1.28908918e-01 -9.57150010e-02 3.63403946e-02 8.10977064e-02
-7.23949376e-02 2.91387907e-02 -1.08475502e-01 1.30374612e-01
-1.13879167e-01 8.04183633e-02 -1.16678698e-02 4.48580346e-02
-8.87779106e-02 6.77491004e-02 -9.87007838e-03 -1.07688287e-03
-4.10958841e-02]

```

第 2 章 ARIMA 模型

Abstract

在本章中我们将首先讲述自回归模型 AR(p)，接着讲述移动平均 MA(q)，最后讲解 ARMA(p,q)，然后将其泛化为 ARIMA(p,d,q)，分别将这些模型用于实际金融时间序列数据拟合。aqt001.py

2 ARIMA 模型

2.1 稳定性和模型选择标准

2.1.1 强稳定性

在我们以前的讨论中，我们说如果一个时间序列各个时间点所对应的随机变量，只要均值和方差不变，就是平稳时间序列。下面我们对强平稳性进行定义。

对于一个时间序列 $\{x_t\}$ ，如果对于 $\forall t_i, m$ ，两个序列： $x_{t_1}, x_{t_2}, \dots, x_{t_N}$ 和 $x_{t_1+m}, x_{t_2+m}, \dots, x_{t_N+m}$ 的统计特性完全相同，则说明该时间序列为强平稳特性。

2.1.2 模型选择标准

我们将用 AIC 来进行模型选择，AIC 的全称为：Akaike Information Criterion，我们通常会选择 AIC 值较小的模型。在实际应用中，还可以使用 BIC 来进行模型选择，BIC 的全称为 Bayes Information Criterion。在本章中我们只用 AIC 来进行模型选择。

假设统计模型的似然函数有 k 参数，最大似然值为 L ，则 AIC 定义为：

$$AIC = -2\log(L) + 2k \quad (24)$$

由式24可知，最大似然值越大或者参数越少，AIC 的值越小，模型就越是好模型。

2.1.3 ADF 检验

在前面所讨论的问题中，我们通常根据自相关系数函数 ACF 和偏自相关系数函数 PACF 来判断稳定性，但是主观性比较强，我们需要一个客观的标准。

我们首先来定义时间序列的阶数，对于下面的非平稳时间序列：

$$x_t = x_{t-1} + w_t \quad (25)$$

其中 $w_t \sim \mathcal{N}(0, \sigma^2)$ 为白噪声信号，且 $x_0 = 0$ ，我们可以得到其均值为：

$$E(x_t) = E(x_{t-1} + w_t) = E(x_{t-1}) + E(w_t) = E(x_{t-1}) = \dots = E(x_0) = 0 \quad (26)$$

同样我们可以得到其方差：

$$Var(x_t) = Var(x_{t-1} + w_t) = Var(x_{t-1}) + Var(w_t) = Var(x_{t-1}) + \sigma^2 = \dots = t\sigma^2 \quad (27)$$

x_t 由于其各时间点对应的随机变量的方差随时间变化，因此不是平稳时间序列。
我们定义 1 阶差分算子：

$$\nabla x_t = Bx_t = x_t - x_{t-1} = w_t \quad (28)$$

对于 Bx_t 为白噪声信号，其显然是平稳时间序列，所以我们称 x_t 为 $I(1)$ 的非平稳时间序列。
我们可以将其定义扩展到 n 阶：

$$Bx_t = x_{t-1} \quad B^2x_t = x_{t-2} \quad B^3x_t = x_{t-3} \quad \dots \quad B^n x_t = x_{t-n} \quad (29)$$

我们还以上面的时间序列 $x_t = x_{t-1} + w_t$ 为例，我们可以将其写为：

$$x_t - x_{t-1} = x_t - Bx_t = (1 - B)x_t = w_t \quad (30)$$

式30中 $1 - B$ 为滞后算子多项式，我们令 $1 - B = 0$ 得出的解为 $B = 1$ ，其为单位根，所以其为非平稳时间序列。这一结论可以推广到更一般的情况，对于如下所示的时间序列：

$$y_t = (1 + \rho)y_{t-1} - \rho y_{t-2} + w_t \quad (31)$$

其所对应的滞后算子多项式为：

$$y_t - (1 + \rho)By_t + \rho B^2y_t = w_t \quad (32)$$

令式32左边为 0，得到的解为： $B = 1$ 和 $B = \frac{1}{\rho}$ ，因为其存在单位根，所以其不是平稳时间序列。

对于任意如下所示时间序列：

$$y_t = \gamma + \rho_1 y_{t-1} + \rho_2 y_{t-2} + \dots + \rho_p y_{t-p} + w_t \quad (33)$$

其中 $w_t \sim \mathcal{N}(0, \sigma^2)$ 为独立同分布 (i.i.d) 噪声信号。可以将式33改写为如下形式：

$$y_t = \gamma + \rho_1 B y_t + \rho_2 B^2 y_t + \dots + \rho_p B^p y_t + w_t \quad (34)$$

将式34右边所有包含 y_t 的项都移到左边，可以得到下式：

$$(1 - \rho_1 B - \rho_2 B^2 - \dots - \rho_p B^p)y_t = \gamma + w_t \quad (35)$$

可以得到其对应的滞后算子多项式方程为：

$$1 - \rho(B) = 1 - \rho_1 B - \rho_2 B^2 - \dots - \rho_p B^p = 0 \quad (36)$$

解这个方程，如果所有解的绝对值均大于 1，则该时间序列为平稳时间序列，如果存在单位根或绝对值小于 1 的根，则其为非平稳时间序列。

以上我们讲解的判断时间序列平稳性的原理，在实际应用中，我们通常采用 ADF 来判断时间序列的平稳性，ADF 模型如下所示：

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \dots + \delta_p \Delta y_{t-p} + w_t \quad (37)$$

其中 α 对应截距， β 对应趋势， $\delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \dots + \delta_p \Delta y_{t-p}$ 为 ADF 的增广项， p 为增广项的期数，其值由 AIC 或 BIC 算法来决定。原假设 H_0 为该序列有单位根是非平稳时间序列： $\gamma = 0$ ；备择假设 H_1 为该序列为平稳时间序列： $\gamma < 0$ 。

这部分原理比较复杂，我们在实际应用中，通常使用 arch 包中的 ADF 函数来完成检验工作，其函数定义为：

$$ADF(y, lags, trend, max_lags, method) \quad (38)$$

其中：

- y ：待判断的时间序列；
- $lags$ ：滞后期数
- $trend$ ：用来控制检验模型的类型
 - 'nc'：不含截距项；
 - 'c'：含截距项；
 - 'ct'：包含截距项和线性趋势项；
 - 'ctt'：包含截距项和线性趋势项以及二次趋势项；
- max_lags ：最大期数
- $method$ ：常用方法为：'aic'、'bic'、't_stat'

下面我们通过一个例子来看怎样使用 ADF 方法，我们以上证综指收益率和收盘价这两个序列为例，我们首先需要安装 python 的 garch 库：

```
1 pip install arch
```

Listing 4: 随机游走拟合上证综指收益率

程序代码如下所示：

```
1 import arch.unitroot as unitroot
2 .....
3 def adf_demo(self):
4     print('ADF检验例程...')
5     data = pd.read_csv(self.data_file, sep='\t', index_col='Trddt')
6     sh_index = data[data.Indexcd==1]
7     sh_index.index = pd.to_datetime(sh_index.index)
8     sh_return = sh_index.Retindex
9     sh_return_adf = unitroot.ADF(sh_return)
10    print(sh_return_adf.summary().as_text())
11    print('stat={0:0.4f}; pvalue={0:0.4f}'.format(sh_return_adf.stat,
12    sh_return_adf.pvalue))
13    print('critical_values:{0}'.format(sh_return_adf.critical_values)
14    )
15    print('1%value={0}'.format(sh_return_adf.critical_values['1%']))
16    if sh_return_adf.stat < sh_return_adf.critical_values['1%']:
17        print('上证综指收益率为平稳时间序列 ^_^')
18    else:
19        print('上证综指收益率为非平稳时间序列 !!!!!!!!!')
20        sh_close = sh_index.Clsindex
21        sh_close_adf = unitroot.ADF(sh_close)
22        if sh_close_adf.stat < sh_close_adf.critical_values['1%']:
23            print('上证综指收盘价为平稳时间序列 ^_^')
24        else:
25            print('上证综指收盘价为非平稳时间序列 !!!!!!!!!')
```

Listing 5: ADF 检验

运行结果如下所示：

Figure 13: 自相关系数函数 ACF

```
量化投资以python为工具
ARMA模型...
ADF检验例程...
Augmented Dickey-Fuller Results
=====
Test Statistic      -7.559
P-value             0.000
Lags                 3
=====
Trend: Constant
Critical Values: -3.45 (1%), -2.87 (5%), -2.57 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
stat=-7.5594; pvalue=-7.5594
critical_values: {'1%': -3.4518314994261337, '5%': -2.8710009653519166, '10%': -2.571810878948318}
1%value=-3.4518314994261337
上证综指收益率为平稳时间序列 ^_^
上证综指收盘价为非平稳时间序列 !!!!!!!!!
```

由上面的结果可以看出，上证综指收益率是稳定的时间序列，收盘价却是不稳定的时间序列。

2.2 自回归模型

2.2.1 背景

我们可以扩展随机游走模型，使当前时间点数据不仅依赖前一时间点的值，同时还依赖前 p 个时间点的值，是这 p 个值的线性组合，这就得到了自回归模型。

2.2.2 模型定义

自回归模型 AR(p) 是随机游走模型的扩展, p 阶自回归模型定义为:

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} + w_t = \sum_{i=1}^p \alpha_i x_{t-i} + w_t \quad (39)$$

其中 $\{w_t\}$ 为白噪声, $\alpha_i \in R$ 且 $\alpha_p \neq 0$ 。

当 $p=1$ 且 $\alpha_1=1$ 时, 自回归模型就退化为随机游走模型。

为后续讨论方便, 我们定义如下运算符:

$$\theta_p(B)x_t = (1 - \alpha_1 B - \alpha_2 B^2 - \dots - \alpha_p B^p)x_t = w_t \quad (40)$$

有了上述模型之后, 我们就可以直接拿来作预测, 如下所示:

$$\begin{aligned} \hat{x}_t &= \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} \\ \hat{x}_{t+1} &= \alpha_1 \hat{x}_t + \alpha_2 x_{t-1} + \dots + \alpha_p x_{t-p+1} \end{aligned} \quad (41)$$

然后依此类推, 可以求出其后 n 个时间点的预测值。

2.2.3 二阶特性

我们首先定义特性方程:

$$\theta_p(B) = 0 \quad (42)$$

解这个方程得到的解的绝对值必须大于 1 才是平稳序列。我们可以举几个实例, 首先是随机游走序列:

随机游走 根据定义 $x_t = x_{t-1} + w_t$ 我们可以得到 $\alpha_1 = 1$, 其特性方程为 $\theta = 1 - B = 0$, 其解为 $B = 1$, 因为其解的绝对值不大于 1, 所以其不是平稳模型。

1 阶自回归 我们假设自回归模型为 $x_t = \frac{1}{4}x_{t-1} + w_t$, 其中 $\alpha_1 = \frac{1}{4}$, 其特性方程为 $\theta = 1 - \frac{1}{4}B = 0$, 其解为 $B = 4$, 该解绝对值大于 1, 所以其是平稳模型。

2 阶自回归模型 我们假设自回归模型为 $x_t = \frac{1}{2}x_{t-1} + \frac{1}{2}x_{t-2} + w_t$, 其特性方程为 $\theta_2(B) = \frac{1}{2}(1-B)(B+2) = 0$, 则其解为 $B = 1, -2$, 其中一个解为单位根, 其绝对值不大于 1, 因此本模型不是平稳模型。虽然本模型不是平稳模型, 但是其他 2 阶自回归模型是完全有可能是平稳模型的。

二阶特性 均值、自协方差、自相关系数定义如下所示:

$$\begin{aligned} \mu_x &= E(x_t) = 0 \\ \gamma_k &= \sum_{i=1}^p \alpha_i \gamma_{k-i}, \quad k > 0 \\ \rho_k &= \sum_{i=1}^p \alpha_i \rho_{k-i}, \quad k > 0 \end{aligned} \quad (43)$$

2.2.4 模拟数据

下面我们来模拟一个 AR(2) 的时间序列, 我们的数据生成和拟合代码如下所示:

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import matplotlib.dates as mdates
5 from matplotlib.font_manager import FontProperties
6 from statsmodels.tsa import stattools
7 from statsmodels.graphics import tsaplots
8 from statsmodels.tsa.arima_model import ARIMA
9
```

```

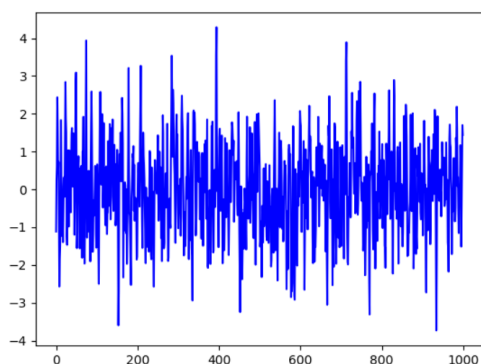
10 class Aqt001(object):
11     def __init__(self):
12         self.name = 'Aqt001'
13
14     def startup(self):
15         print('ARMA模型...')
16         self.simulate_ar2()
17
18     def simulate_ar2(self):
19         print('模拟AR(2)')
20         alpha1 = 0.666
21         alpha2 = -0.333
22         wt = np.random.standard_normal(size=1000)
23         x = wt
24         for t in range(2, len(wt)):
25             x[t] = alpha1 * x[t-1] + alpha2 * x[t-2] + wt[t]
26         plt.plot(x, c='b')
27         plt.show()
28         ar2 = stattools.ARMA(x, (2, 0)).fit(disp=False)
29         print('p={0} **** {1}; q={2}***{3}; {4} - {5} - {6}'.format(
30             ar2.k_ar, ar2.arparams, ar2.k_ma, ar2.maparams,
31             ar2.aic, ar2.bic, ar2.hqic)
32         )
33         arima2_0_0 = ARIMA(x, order=(2, 0, 0)).fit(disp=False)
34         print('ARIMA: p={0} **** {1}; q={2}***{3}; {4} - {5} - {6}'. \
35             format(arima2_0_0.k_ar, arima2_0_0.arparams,
36                 arima2_0_0.k_ma, arima2_0_0.maparams,
37                 arima2_0_0.aic, arima2_0_0.bic,
38                 arima2_0_0.hqic)
39         )
40         resid = arima2_0_0.resid
41         # 绘制ACF
42         acfs = stattools.acf(resid)
43         print(acfs)
44         tsaplots.plot_acf(resid, use_vlines=True, lags=30)
45         plt.title('ACF figure')
46         plt.show()
47         pacfs = stattools.pacf(resid)
48         print(pacfs)
49         tsaplots.plot_pacf(resid, use_vlines=True, lags=30)
50         plt.title('PACF figure')
51         plt.show()

```

Listing 6: AR 数据模拟和拟合示例

我们首先生成一个 1000 个数据点的均值为 0 方差为 1 的白噪声数据，然后根据 $x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + w_t = 0.666 \times x_{t-1} - 0.333 \times x_{t-2} + w_t$ 公式，生成拟合数据。我们绘制该模拟数据图像，接着我们分别用 ARMA 和 ARIMA 进行拟合，求出系数，然后计算出模拟选择的参数：AIC、BIC、HQIC 的值，最后我们求出模型拟合的残差，并绘制出残差自相关系数函数 ACF 和偏自相关系数函数 PACF 的图像。模拟数据图像为：

Figure 14: AR2 模拟生成数据图



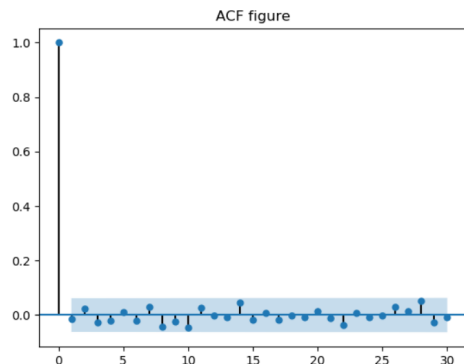
无论是 ARMA 还是 ARIMA 拟合，我们都可以得到较为正确的数据，同时残差的 ACF 和 PACF 也表明其是白噪声序列，运行结果如下所示：

Figure 15: 程序运行结果

```
量化投资以python为工具
ARMA模型...
模拟AR(2)
p=2 **** [ 0.67734879 -0.30934048]; q=0***[]; 2805.45882423338 - 2825.0898453493082 - 2812.919982104708
ARIMA: p=2 **** [ 0.67734879 -0.30934048]; q=0***[]; 2805.45882423338 - 2825.0898453493082 - 2812.919982104708
[ 1.00000000e+00 -1.27594914e-02 2.33980378e-02 -2.83973503e-02
-1.99624524e-02 1.17663389e-02 -1.95196686e-02 2.85292757e-02
-4.28202635e-02 -2.52670740e-02 -4.48898555e-02 2.56210627e-02
-3.07874134e-03 -6.99614103e-03 4.61211026e-02 -1.70347399e-02
6.79310491e-03 -1.76508823e-02 -3.03059320e-03 -7.93521381e-03
1.47847033e-02 -1.00313054e-02 -3.58293988e-02 6.74092169e-03
-8.77538910e-03 -4.66989248e-04 2.93222689e-02 1.34644761e-02
5.12237137e-02 -2.72023000e-02 -8.38201436e-03 -2.17939430e-02
-5.25994727e-02 -4.20756414e-02 -1.87352437e-02 1.33258139e-02
6.19913054e-03 4.16316144e-03 1.61043531e-02 -1.14240579e-02
-1.32598149e-02]
[ 1.
-0.01277226 0.0232856 -0.0279126 -0.02130531 0.0126565
-0.01921062 0.0265783 -0.04146117 -0.02846307 -0.04359029 0.02540431
-0.00479445 -0.00996858 0.04490654 -0.01295893 0.00138853 -0.01396058
-0.00845372 -0.00942079 0.01628618 -0.01232774 -0.03515738 0.00746495
-0.00265584 -0.00863967 0.03102144 0.01237213 0.04962798 -0.02321584
-0.01278117 -0.02126148 -0.05607669 -0.04583023 -0.01970885 0.01196684
0.01507584 0.00205992 0.01972234 -0.01629268 -0.02133968]
```

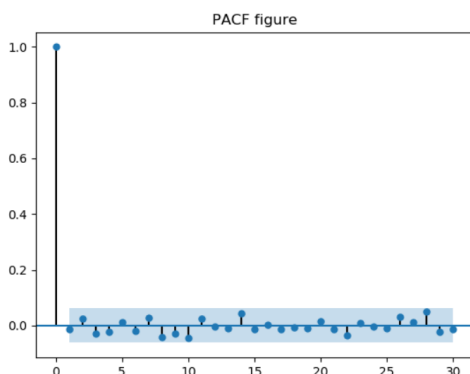
残差的自相关系数函数 ACF 图：

Figure 16: 残差的自相关系数函数 ACF 图



残差的偏自相关系数函数 PACF 图：

Figure 17: 残差的偏自相关系数函数 PACF 图



由 ACF 和 PACF 图可以看出，我们残差是比较典型的随机白噪声序列，由此可见我们拟合还是很好的。

2.2.5 展望

在理解了基本理论之后，我们将引入最终的 ARIMA 模型，并用 ARIMA 模型来拟合真实上证综指收盘价时间序列，并用我们的拟合模型来预测最后 5 日的收盘价，在这个实际例子中，看我们模型的表现如何。

2.3 ARIMA 模型

2.3.1 背景

我们不仅可以对当前时间点数值对之前时间点数值进行建模，我们也可以对前面时间点随机噪声对当前时间点的影响进行建模，同时由于原始信号可能非常不平稳，但是我们求出其差值序列后，可能就变为平稳序列了，这就是 ARIMA 模型要解决的问题。为了讨论 ARIMA 模型，我们首先介绍差分的概念：

$$\begin{aligned} \{x_1, x_2, \dots, x_N\} \\ \{d_1^1, d_2^1, \dots, d_{N-1}^1\} &= \{x_2 - x_1, x_3 - x_2, \dots, x_N - x_{N-1}\} \\ \{d_1^2, d_2^2, \dots, d_{N-1}^2\} &= \{d_2^1 - d_1^1, d_3^1 - d_2^1, \dots, d_{N-1}^1 - d_{N-2}^1\} \end{aligned} \quad (44)$$

上式中分别为原始时序信号，然后是一阶差分和二阶差分。

2.3.2 模型选择标准

BIC 定义 我们已经介绍过一个模型选择标准 AIC (Akaike Information Criterion)，其会惩罚参数多的模型，因为这些模型容易产生过拟合 (Overfitting)。接下来我们要介绍另一个模型选择参数 BIC (Bayes Information Criterion)，与 AIC 相比，其会更倾向于惩罚参数多的模型，同样是值越小越好，BIC 定义如下所示：

$$BIC = -2 \log(L) + k \log N \quad (45)$$

其中 L 为似然函数的最大值，k 为模型的参数，N 为数据点个数。

Ljung-Box 检测 我们的缺省假设 H_0 为：对于一个拟合的时序信号，对所有滞后时点 lags，都是独立同分布 (i.i.d) 的，即不存在相关性。

备择假设 H_a 为：这些信号不是独立同分布 (i.i.d) 的，具有相关性。

我们定义统计量 Q：

$$Q = n(n+2) \sum_{k=1}^h \frac{\hat{\rho}_k^2}{n-k} \quad (46)$$

式 46 中 n 为时间序列长度， h 为最大滞后期数， ρ_k 第 k 自相关系数。Ljung-Box 检测原理比较复杂，但是在 python 语言中，经过运算可以求出 Q 值，以及大于 Q 值的概率，实际上我们看 1~12 滞后期的 Q 值和大于 Q 值的概率，如果该概率小于显著水平如 0.05 时，就拒绝缺省假设 (不存在相关性)，选择备择假设，否则反之。

2.3.3 定义

同时考虑之前时间点的信号和噪声值，我们就可以得到如下 ARMA 模型：

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} + w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \dots + \beta_q w_{t-q} \quad (47)$$

如果我们对欲研究的信号求出一阶或二阶差分，然后再利用式47的模型，就是 ARIMA 模型了。其中 $\{w_t\}$ 为白噪声，其均值为 0，方差为 σ^2 。

其特性方程可以表示为：

$$\theta_p(B)x_t = \phi_q(B)w_t \quad (48)$$

由上面的讨论可以看出，AR(p) 和 MA(q) 都是 ARIMA 模型的特殊情况，在同样精度的条件下，ARIMA 模型所需参数最小。

2.3.4 数据仿真

在理解了 ARIMA 模型定义之后，我们来模拟一下 ARIMA 过程。假设我们要模拟的 ARIMA 模型为：

$$\begin{aligned} x_t &= \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} \\ &= 1.2 \times x_{t-1} - 0.7 \times x_{t-2} + w_t - 0.06 \times w_{t-1} - 0.02 \times w_{t-2} \end{aligned} \quad (49)$$

生成模拟数据并利用 ARIMA 拟合的程序如下所示：

```
1 def simulate_arma_p_d_q(self):
2     print('模拟ARIMA(p,d,q)过程')
3     np.random.seed(8)
4     alpha1 = 1.2
5     alpha2 = -0.7
6     beta1 = -0.06
7     beta2 = -0.02
8     w = np.random.standard_normal(size=1000)
9     x = w
10    for t in range(2, len(w)):
11        x[t] = alpha1 * x[t-1] + alpha2*x[t-2] + w[t] + beta1 * w[t
12    -1] + beta2*w[t-2]
13    plt.plot(x, c='b')
14    plt.title('ARIMA(p, d, q) Figure')
15    plt.show()
16    # 查看ACF
17    acfs = stattools.acf(x)
18    print('ARIMA(q,d,q) ACFS:\r\n{0}'.format(acfs))
19    tsaplots.plot_acf(x, use_vlines=True, lags=30)
20    plt.title('ARIMA(p,d,q) ACF')
21    plt.show()
22    # ARIMA拟合
23    min_ABQIC = sys.float_info.max
24    arima_model = None
25    break_loop = False
26    for p in range(0, 5):
27        if break_loop:
28            break
29        for q in range(0, 5):
30            print('try {0}, d, {1}...'.format(p, q))
31            try:
32                arima_p_d_q = ARIMA(x, order=(p, 0, q)).fit(disp=
33    False)
34                print('..... fit ok')
35                if arima_p_d_q.aic < min_ABQIC:
36                    print('..... record good model')
37                    min_ABQIC = arima_p_d_q.aic
38                    arima_model = arima_p_d_q
39                    #if l==p and l==q:
40                        # break_loop = True
41            except Exception as ex:
42                print('.....!!!!!! Exception')
```



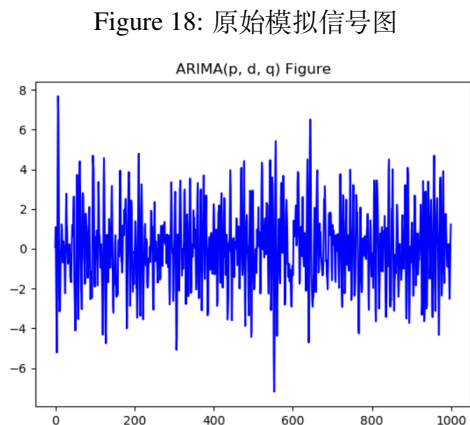
```

42     print('ARIMA: p={0} **** {1}; q={2}***{3}; {4} - {5} - {6}'. \
43           format(arima_model.k_ar, arima_model.arparams,
44                 arima_model.k_ma, arima_model.maparams,
45                 arima_model.aic, arima_model.bic,
46                 arima_model.hqic)
47     ),
48     ),
49     arima_model = ARIMA(x, order=(2, 0, 2)).fit(dis= False)
50     print('God_View:ARIMA: p={0} **** {1}; q={2}***{3}; {4} - {5} -
51           {6}'. \
52           format(arima_model.k_ar, arima_model.arparams,
53                 arima_model.k_ma, arima_model.maparams,
54                 arima_model.aic, arima_model.bic,
55                 arima_model.hqic)
56     )
57     resid = arima_model.resid
58     # 绘制ACF
59     acfs = stattools.acf(resid)
60     print(acfs)
61     tsaplots.plot_acf(resid, use_vlines=True, lags=30)
62     plt.title('ARIMA(p,d,q) ACF figure')
63     plt.show()
64     pacfs = stattools.pacf(resid)
65     print(pacfs)
66     tsaplots.plot_pacf(resid, use_vlines=True, lags=30)
67     plt.title('ARIMA(p,d,q) PACF figure')
68     plt.show()

```

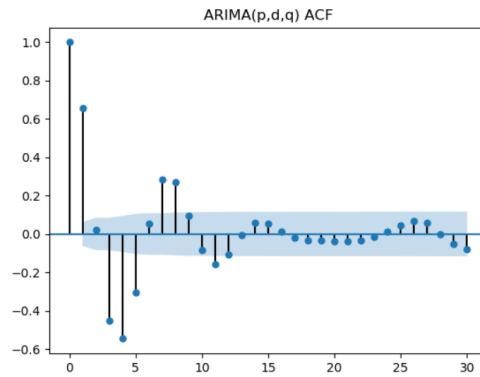
Listing 7: AR 数据模拟和拟合示例

生成的时序信号 x 为:



该信号的自相关系数函数 ACF 图为:

Figure 19: 原始模拟信号 ACF 图



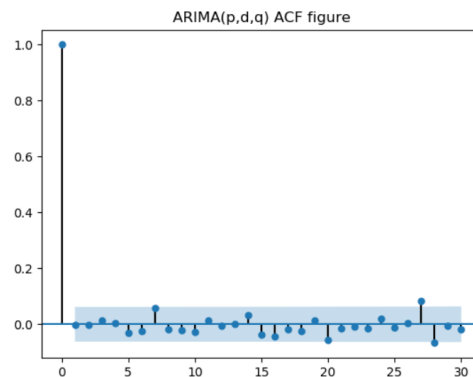
由图中可以看出，该信号具有非常强的自相关性。
接着我们用 ARIMA 模型来模拟该信号，上面程序注释部分为求最佳 ARIMA 模型的 p 和 q 参数，以 AIC 作为模型选择标准，因为我们知道模型为 ARIMA(2,0,2)，所以我们同时也用 ARIMA(2,0,2) 来进行拟合，程序运行结果如下所示：

Figure 20: 程序运行结果

```
量化投资以python为工具
模拟ARIMA(p, d, q)过程
ARIMA(q, d, q) ACFS:
[ 1.00000000e+00  6.56463444e-01  2.20934132e-02 -4.52132579e-01
-5.46049881e-01 -3.05649369e-01  5.20559164e-02  2.84818841e-01
 2.72038628e-01  9.50275817e-02 -8.62933103e-02 -1.56204731e-01
-1.08730271e-01 -8.02642608e-03  6.07353814e-02  5.61043349e-02
 1.44722310e-02 -1.90736312e-02 -3.20308294e-02 -3.33016162e-02
-3.95106929e-02 -3.89611615e-02 -3.26327450e-02 -1.67104910e-02
 1.27078752e-02  4.52637219e-02  6.97142822e-02  5.86791384e-02
 8.25996402e-05 -5.20741287e-02 -7.86355496e-02 -7.13731430e-02
-3.41553392e-02  1.99413935e-02  6.02743403e-02  7.28214056e-02
 4.41942956e-02 -8.77721515e-03 -3.81945748e-02 -2.65299912e-02
 8.97896382e-03]
God_View:ARIMA: p=2 **** [ 1.16735279 -0.72626584]; q=2***[-0.05184418 -0.06459538]; 2890.9503824872345 - 2920.396914161
1274 - 2902.142119294227
[ 1.00000000e+00 -7.27774367e-04 -1.11101018e-03  1.28634523e-02
 5.48318944e-03 -3.03183072e-02 -2.43067634e-02  5.81369511e-02
-1.67805871e-02 -2.03352340e-02 -2.89272450e-02  1.43722350e-02
-4.98312176e-03  2.17342132e-03  3.36256042e-02 -3.689113979e-02
-4.26800868e-02 -1.72822287e-02 -2.50926413e-02  1.28230095e-02
-5.63492734e-02 -1.60329820e-02 -8.58773643e-03 -1.42536089e-02
 1.91587444e-02 -1.10064043e-02  4.93949871e-03  8.31719158e-02
-6.68677052e-02 -5.70060946e-03 -1.79923901e-02 -1.18040998e-02
-3.38040533e-02  2.36494039e-02  1.66539078e-02  3.31933370e-02
 5.39490149e-02 -6.82510840e-03  2.23837826e-03  3.81197763e-02
 3.76186214e-03]
```

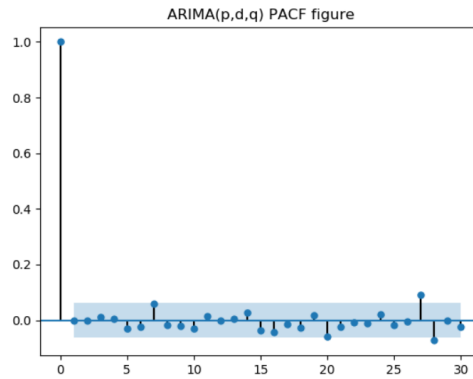
接着我们求出残差序列，残差序列自相关系数函数 ACF 图如下所示：

Figure 21: 残差自相关系数函数 ACF 图



残差序列偏自相关系数函数图 PACF 如下所示：

Figure 22: 残差偏自相关系数函数 ACF 图



由图中可以看出，残差序列基本上是白噪声信号。

2.3.5 金融数据拟合及预测

接下来我们用 ARIMA 模型，来拟合上证综指收盘价，我们利用除最后 3 天的数据来得出 ARIMA 模型，然后利用拟合出的 ARIMA 模型来预测后 3 天的收盘价，来看我们模型的性能。根据经验，对于股票的收盘数据来说，采用取对数后再求 1 阶差分的形式，可以取得更好的效果，因此我们会先对数据进行预处理，然后再来用 ARIMA 模型来拟合数据。程序如下所示：

```

1  def arima_demo(self):
2      register_matplotlib_converters()
3      data = pd.read_csv(self.data_file, sep='\t', index_col='Trddt')
4      sh_index = data[data.Indexcd==1]
5      sh_index.index = pd.to_datetime(sh_index.index)
6      raw_data = sh_index.Clsindex
7      train_data = raw_data[:-3]
8      close_price = np.log(train_data)
9      plt.plot(close_price)
10     plt.show()
11     print(train_data.head(n=3))
12     # ARIMA拟合
13     min_ABQIC = sys.float_info.max
14     arima_model = None
15     for p in range(0, 5):
16         for q in range(0, 5):
17             print('try {0}, d, {1}...'.format(p, q))
18             try:
19                 arima_p_d_q = ARIMA(close_price, order=(p, 1, q)).fit
20                 (disp=False)
21                 print('..... fit ok')
22                 if arima_p_d_q.aic < min_ABQIC:
23                     print('..... record good model')
24                     min_ABQIC = arima_p_d_q.aic
25                     arima_model = arima_p_d_q
26             except Exception as ex:
27                 print('.....!!!!!! {0}'.format(ex))
28     print('ARIMA: p={0} **** {1}; q={2}***{3}; {4} - {5} - {6}'. \
29           format(arima_model.k_ar, arima_model.arparams,
30                 arima_model.k_ma, arima_model.maparams,
31                 arima_model.aic, arima_model.bic,
32                 arima_model.hqic)
33     )
34     resid = arima_model.resid
35     # 绘制ACF
36     acfs = stattools.acf(resid)
37     print(acfs)

```

```

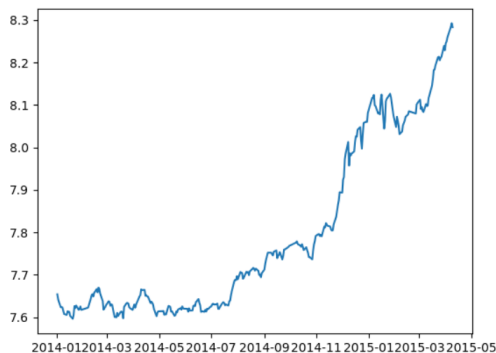
37     tsaplots.plot_acf(resid, use_vlines=True, lags=30)
38     plt.title('ARIMA(p,d,q) ACF figure')
39     plt.show()
40     pacfs = stattools.pacf(resid)
41     print(pacfs)
42     tsaplots.plot_pacf(resid, use_vlines=True, lags=30)
43     plt.title('ARIMA(p,d,q) PACF figure')
44     plt.show()
45     # ADF检验
46     resid_adf = unitroot.ADF(resid)
47     print('stat={0:0.4f} vs 1%_cv={1:0.4f}'.format(resid_adf.stat,
48     resid_adf.critical_values['1%']))
49     if resid_adf.stat < resid_adf.critical_values['1%']:
50         print('resid为稳定时间序列 ^^')
51     else:
52         print('resid为非稳定时间序列!!!!')
53     # Ljung-Box检验
54     resid_ljung_box = stattools.q_stat(stattools.acf(resid)[1:12],
55     len(resid))
56     resid_lbv = resid_ljung_box[1][-1]
57     print('resid_ljung_box_value={0}'.format(resid_lbv))
58     # 0.05为显著性水平
59     if resid_lbv < 0.05:
60         print('resid为平稳时间序列 ^^')
61     else:
62         print('resid为非平稳时间序列!!!!!!')
63     # 预测
64     y = arima_model.forecast(3)[0] #(len(train_data), len(raw_data),
65     dynamic=True)
66     print('预测值: {0}'.format(np.exp(y)))
67     print('row_data:{0}'.format(raw_data))
68     print('train_data:{0}'.format(train_data))

```

Listing 8: ARIMA 数据拟合上证综指收盘价

我们首先绘制出上证综指收盘价曲线：

Figure 23: 上证综指收盘价



接着我们对该数据经过对数差分后，利用 ARIMA 来进行拟合，得到拟合模型为：

Figure 24: 拟合后的 ARIMA 模型

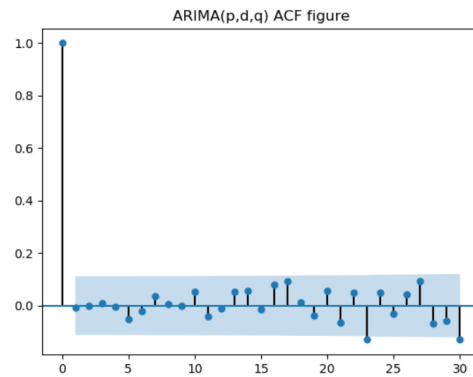
```

ARIMA: p=0 **** [ ]: q=4***[ 0.0533824 -0.01120743 -0.06658252 0.15689616]; -1807.0408546562194 - -1784.6797681706962
-1798.0988716797663
[ 1.
-0.00731921 -0.00178857 0.00868245 -0.00274171 -0.05269537
-0.02056405 0.03496425 0.0073848 -0.00166092 0.05352974 -0.04213294
-0.01009974 0.05372638 0.05711811 -0.01300352 0.08149255 0.0926249
0.0132617 -0.03754272 0.0548461 -0.06588346 0.04993921 -0.12993628
0.04927725 -0.03086053 0.04413533 0.09256284 -0.06752662 -0.05814295
-0.12733782 0.0336995 -0.03802792 0.07169979 0.07038559 0.04172848
-0.03240709 0.02693128 -0.03516964 -0.04603044 -0.0568947 ]

```

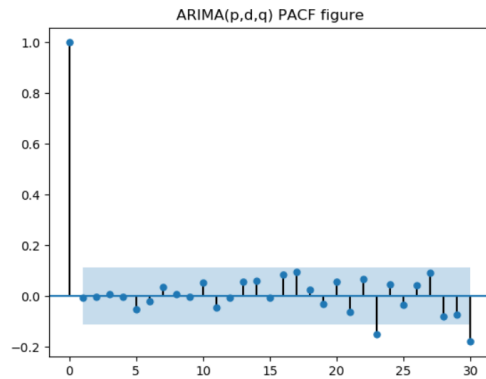
残差序列的自相关系数函数 ACF 图为：

Figure 25: 残差序列的自相关系数函数 ACF 图



残差序列的偏自相关系数函数 PACF 图为：

Figure 26: 残差序列的偏自相关系数函数 PACF 图



进行 ADF 检验的结果为：

Figure 27: ADF 检验的结果

```
stat=-17.5787 vs 1%_cv=-3.4519
resid为稳定时间序列
```

进行 Ljung-Box 检验结果为：

Figure 28: Ljung-Box 检验结果

```
resid_ljung_box_value=0.9915215214530787
resid为非平稳时间序列!!!!!!
```

最后我们拿我们的模型进行预测，后三天的预测结果为：

Figure 29: ARIMA 模型预测后三天结果

```
预测值: [3958.4030514 3982.14296761 3990.28084529]
```

实际值为：

Figure 30: 实际收盘价

2015-04-09	3957.534
2015-04-10	4034.310
2015-04-13	4121.715
2015-04-14	4135.565

我们看到，我们的模型基本预测出了后三天的连涨行情，只不过上涨的幅度有一些小。

第3章 GARCH 模型

Abstract

在本章中我们将首先讲述条件异方差模型 GARCH (Generalized AutoRegressive Conditional Heteroskedastic)，并将 GARCH 模型用于实际金融时间序列数据拟合。aqt002.py

3 GARCH 模型

python 的 ARIMA: <https://www.colabug.com/3933896.html> Ln125 Advanced Algorithmic Trading

4 汇总

f000022 c000008 e000050

参考文献: [I.MLearning \[1999\]](#)—[A.NikolaosAI \[1999\]](#)—[Bakry et al. \[2015\]](#)

References

- Andreas Nikolaos A.NikolaosAI. *A Book He Wrote*. His Publisher, Erewhon, NC, 1999.
- Amr Bakry, Mohamed Elhoseiny, Tarek El-Gaaly, and Ahmed M. Elgammal. Digging deep into the layers of cnns: In search of how cnns achieve view invariance. *CoRR*, abs/1508.01983, 2015. URL <http://arxiv.org/abs/1508.01983>.
- Ivan Marc I.MLearning. Some related article I wrote. *Some Fine Journal*, 99(7):1–100, January 1999.