Chapter-2

Exercise-1

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14
四级考	98	85	89	84	81	70	92	67	84	80	60	81	73	70
六级考	90	82	88	80	82	66	88	68	84	77	64	79	75	73

1.1

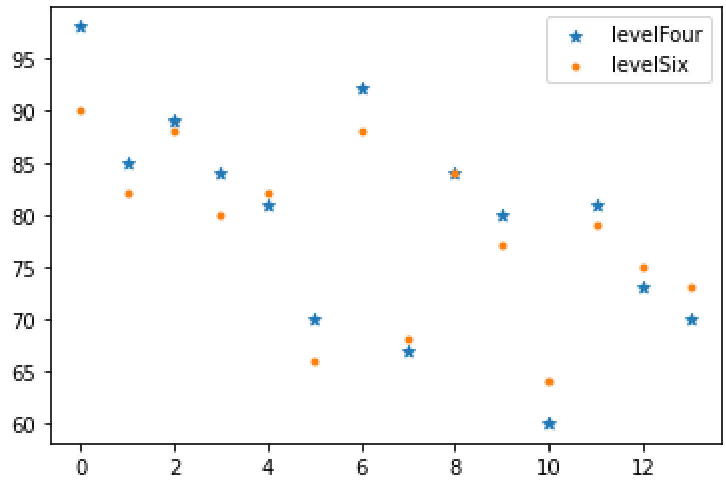
求出这两者之间的相关系数，画出它们的散点图；

代码：

```
import numpy as np
import pandas as pd

levelFour = np.array([98,85,89,84,81,70,92,67,84,80,60,81,73,70])
levelSix = np.array([90,82,88,80,82,66,88,68,84,77,64,79,75,73])
data = pd.DataFrame({
    'levelFour':levelFour,
    'levelSix':levelSix
})

# 求出这两者之间的相关系数
print(data.corr())
# 画出它们的散点图
from matplotlib import pyplot as plt
y = np.arange(0,levelFour.size,1)

plt.scatter(y,levelFour,marker='*',label='levelFour')
plt.scatter(y,levelSix,marker='.',label='levelSix')
plt.show()
```

运行：

```
           levelFour  levelSix
levelFour   1.000000  0.964148
levelSix    0.964148  1.000000
```



解释：
两者的相关系数为`0.964148`。

1.2

求出各考试的均值、标准差、偏度、峰度；
代码：

```
import numpy as np
import pandas as pd

levelFour = np.array([98,85,89,84,81,70,92,67,84,80,60,81,73,70])
levelSix = np.array([90,82,88,80,82,66,88,68,84,77,64,79,75,73])
data = pd.DataFrame({
    'levelFour':levelFour,
    'levelSix':levelSix
})

for k in ['levelFour','levelSix']:
    print(k)
    print('均值' + ' ' *3 + str(data[k].mean()))
    print('标准差' + ' ' * 3 + str(data[k].std()))
    # 偏度
    print('偏度' + ' ' * 3 + str(data[k].skew()))
    # 峰度
    print('峰度' + ' ' * 3 + str(data[k].kurt()))
    print('\n')
```

运行：

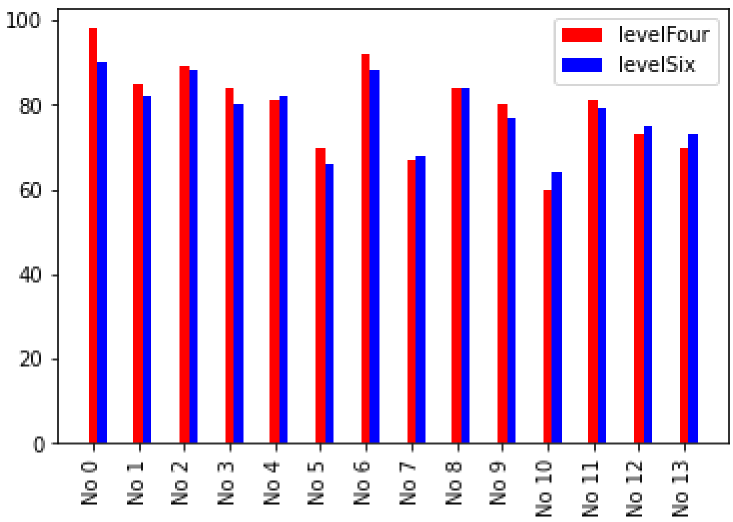
```
levelFour
均值   79.57142857142857
标准差   10.463962227306896
偏度   -0.1695617201421985
峰度   -0.3796252326410898

levelSix
均值   78.28571428571429
标准差   8.27813218833044
偏度   -0.36511259060435064
峰度   -0.8538934689358708
```

1.3

画出各考试的盒形图（在一张图上的对比盒形图）
代码：

```
import numpy as np
import pandas as pd

levelFour = np.array([98,85,89,84,81,70,92,67,84,80,60,81,73,70])
levelSix = np.array([90,82,88,80,82,66,88,68,84,77,64,79,75,73])
data = pd.DataFrame({
    'levelFour':levelFour,
    'levelSix':levelSix
})

from matplotlib import pyplot as plt

xFour = range(0,levelFour.size)
xSix = [i + 0.2 for i in range(0,levelFour.size)]
x_lalel = ['No '+ format(i) for i in xFour]
plt.bar(xFour,levelFour,width=0.2,color='red',label='levelFour')
plt.bar(xSix,levelSix,width=0.2,color='blue',label='levelSix')
plt.xticks(xFour,x_lalel,rotation=90)
plt.legend(loc = 'upper right')
plt.show()
```

运行：



1.4

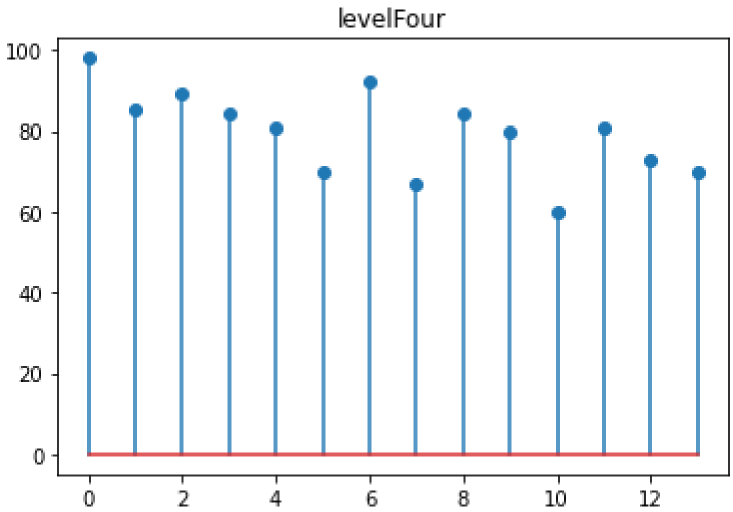
画出各考试的茎叶图。

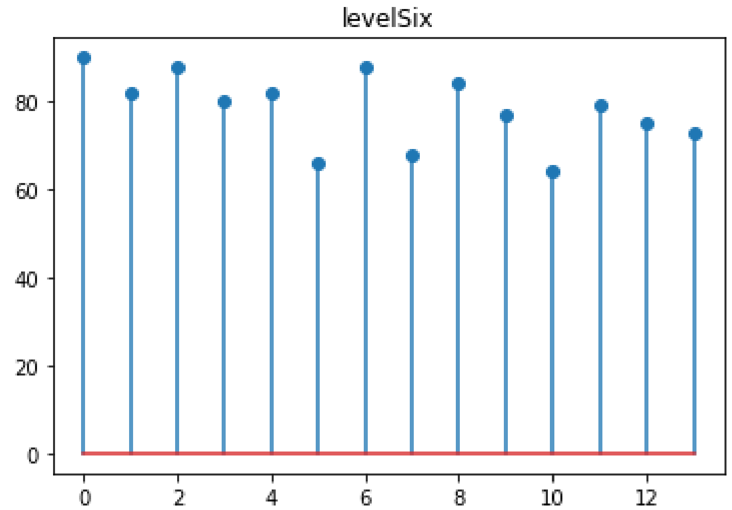
代码：

```
import numpy as np
import pandas as pd

levelFour = np.array([98,85,89,84,81,70,92,67,84,80,60,81,73,70])
levelSix = np.array([90,82,88,80,82,66,88,68,84,77,64,79,75,73])
```

```
data = pd.DataFrame({
    'levelFour':levelFour,
    'levelSix':levelSix
})

from matplotlib import pyplot as plt
x = np.arange(0,levelFour.size,1)

plt.stem(x,levelFour)
plt.title('levelFour')
plt.show()
plt.stem(x,levelSix)
plt.title('levelSix')
plt.show()
```

运行：





Exercise-2

某种电子元件的平均寿命x(单位：小时)服从正态分布，现测得16只元件的寿命分别为159、280、101、212、224、379、179、264、222、362、168、149、260、485、170，试求元件寿命的极大似然估计及区间估计，并问有否理由认为元件的平均寿命地大于225小时（α=0.05）。

代码：

```
import pandas as pd

dataSer =
pd.Series([159,280,101,212,224,379,179,264,222,362,168,149,260,485,170])

# 极大似然估计
print('极大似然估计')
mean,std = stats.norm.fit(dataSer)
print('平均值=', mean)
print('标准差= ', std)

print('\n')

# 区间估计
print('区间估计')
from scipy import stats
pop_mean=225
t,p_two = stats.ttest_1samp(dataSer,pop_mean)
print('t值=',t,'双尾检验的P值',p_two)

alpha = 0.05

if(p_two < alpha):
    print('元件的平均寿命小于225小时')
else:
    print('元件的平均寿命大于225小时')
```

运行：

```
极大似然估计
平均值= 240.93333333333334
标准差=  98.69985928167385


区间估计
t值= 0.6040239043549231 双尾检验的P值 0.5554898769672549
元件的平均寿命大于225小时
```

Exercise-3

考虑从2001年9月1日到2011年9月30日美国运通公司（AXP）、CRSP价值权重指数（VW）、CRSO的等权重指数（EW）以及S&P综合指数的日简单收益率。收益率中包含有支付的股息。数据来自d-axp3dx-0111.txt.对AXP股票的对数收益的均值等于零的原假设进行检验。在5%的显著性水平下，得出你的结论。
代码：

```
import pandas as pd
import math

df = pd.read_csv('d-axp3dx-0111.txt',delim_whitespace = True)

df['axp_log'] = df.apply(lambda x: math.log(x['axp'] + 1), axis=1)

from scipy import stats
```

```
pop_mean=math.log(1)
t,p_two = stats.ttest_1samp(df['axp_log'],pop_mean)
print('t值=',t,'双尾检验的P值',p_two)

alpha = 0.05

if(p_two < alpha):
    print('AXP股票的对数收益的均值等于零')
else:
    print('AXP股票的对数收益的均值不等于零')
```

运行：

```
t值= 0.35998718143540415 双尾检验的P值 0.7188867193267378
AXP股票的对数收益的均值不等于零
```

Exercise-4

考虑从1940年1月到2011年9月S&P综合指数的月股票收益率（m-ge3dx-4011.txt），进行下面的检验，在5%的显著性水平下得出你的结论。

4.1

检验假设H0: μ=0，其备择假设为H1：μ<>0，这里μ是收益率的均值；
代码：

```
import pandas as pd

df = pd.read_csv('m-ge3dx-4011.txt',delim_whitespace = True)

from scipy import stats
pop_mean=0
t,p_two = stats.ttest_1samp(df['sp'],pop_mean)
print('t值=',t,'双尾检验的P值',p_two)

alpha = 0.05

if(p_two < alpha):
    print('SP股票的收益的均值不等于零')
else:
    print('SP股票的收益的均值等于零')
```

运行：

```
t值= 4.243789356353936 双尾检验的P值 2.436583365338209e-05
SP股票的收益的均值不等于零
```

4.2

检验假设H0: m3=0，其备择假设为H1：m3<>0，这里m3是收益率的偏度；
代码：

```
import pandas as pd

df = pd.read_csv('m-ge3dx-4011.txt',delim_whitespace = True)
from scipy import stats
p = stats.skewtest(df['sp'])[1]
print('p值',p)
alpha = 0.05
if p < alpha:
    print('SP股票的收益率偏度不等于0')
else:
    print('SP股票的收益率偏度等于0')
```

运行：

```
p值 3.275294917526283e-11
SP股票的收益率偏度不等于0
```

4.3

检验假设H0: K=3，其备择假设为H1：K<>0，这里K是收益率的峰度；
代码：

```
import pandas as pd

df = pd.read_csv('m-ge3dx-4011.txt',delim_whitespace = True)
from scipy import stats
p = stats.kurtosistest(df['sp'])[1]
print('p值',p)
alpha = 0.05
if p < alpha:
    print('SP股票的收益率峰度不等于3')
else:
    print('SP股票的收益率峰度等于3')
```

运行：

```
p值 4.992725622126931e-13
SP股票的收益率峰度不等于3
```

Exercise-5

从芝加哥的联邦储备银行得到日汇率，数据是经过纽约联邦储备银行认证的纽约市每日中午买入价。考虑从2007年1月2日到2011年11月30日美元对英镑、日元的汇率（数据见：d-fx-usjp-0711.txt）。

5.1

每个汇率的日对数收益率；
代码：

```python
import pandas as pd
import numpy as np
import math

df = pd.read_csv('d-fx-usjp-0711.txt',delim_whitespace = True)
df['rate_log'] = df.apply(lambda x: math.log(x['rate'],math.e), axis=1)

print(np.diff(df['rate_log']))
```

运行：

```
[ 0.0062917  -0.00251193 -0.00428482 ...  0.00500932 -0.0024373
 -0.00360268]
```

5.2

汇率的日对数收益率的样本均值、标准差、偏度、超额峰度、最大值和最小值；
代码：

```python
import pandas as pd
import numpy as np
import math

df = pd.read_csv('d-fx-usjp-0711.txt',delim_whitespace = True)
df['rate_log'] = df.apply(lambda x: math.log(x['rate'],math.e), axis=1)

diffDf = pd.DataFrame({
    'diff':np.diff(df['rate_log'])
})

print('均值' + ' ' *3 + str(diffDf['diff'].mean()))
print('标准差' + ' ' * 3 + str(diffDf['diff'].std()))
# 偏度
print('偏度' + ' ' * 3 + str(diffDf['diff'].skew()))
# 峰度
print('峰度' + ' ' * 3 + str(diffDf['diff'].kurt()))
print('最大值' + ' ' * 3 + str(diffDf['diff'].max()))
print('最小值' + ' ' * 3 + str(diffDf['diff'].min()))
```

运行：

```
均值   -0.0003446921894018703
标准差   0.0075146100501606965
偏度   -0.4175117907034683
峰度   4.873053676621373
最大值   0.030592852978797325
最小值   -0.05215647959436165
```

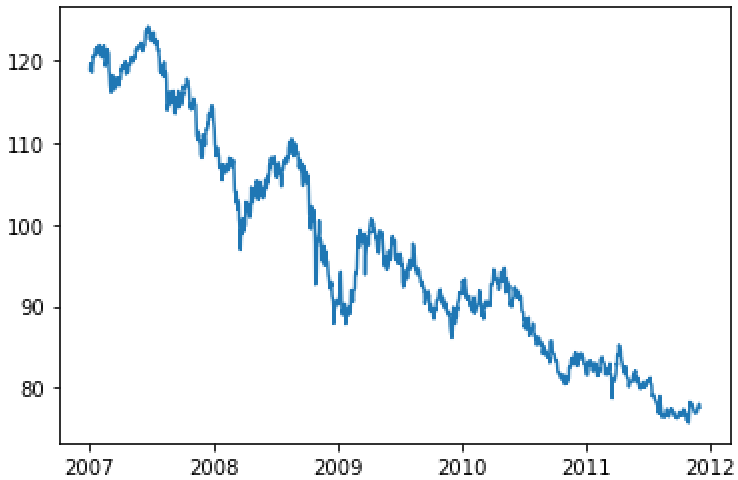
5.3

将作图区间分为两个，在其中画出美元/日元汇率的走势图及日对数收益率的密度函数（包括核密度估计和正态密度估计）

```python
import pandas as pd
```

```
# 导入 matplotlib.pyplot，用以画图
from matplotlib import pyplot as plt

df = pd.read_csv('d-fx-usjp-0711.txt',delim_whitespace = True)

df['year'] = df['year'].astype('str')
df['mon'] = df['mon'].astype('str')
df['day'] = df['day'].astype('str')

df['date'] = pd.to_datetime(df['year'].str.cat([df['mon'],df['day']],sep='-'))


x = df[['date']].values
y = df[['rate']].values

plt.plot(x,y)
plt.show()
```

运行：



5.4

检验假设H0: μ=0，其备择假设为H1：μ<>0，这里μ表示美元/日元汇率的日对数收益率的均值，在5%的显著性水平下给出你的结论。
代码：

```
import pandas as pd
import numpy as np
import math

df = pd.read_csv('d-fx-usjp-0711.txt',delim_whitespace = True)

df['rate_log'] = df.apply(lambda x: math.log(x['rate'],math.e), axis=1)

diffDf = pd.DataFrame({
    'diff':np.diff(df['rate_log'])
})

from scipy import stats
pop_mean=0
print(stats.ttest_1samp(diffDf['diff'],pop_mean))
```

运行：

```
Ttest_1sampResult(statistic=-1.6132803474573114, pvalue=0.10693887848638878)
```

解释：
p-value == 0.1069 > 0.05，故无法拒绝原假设，可得μ == 0

Exercise-6

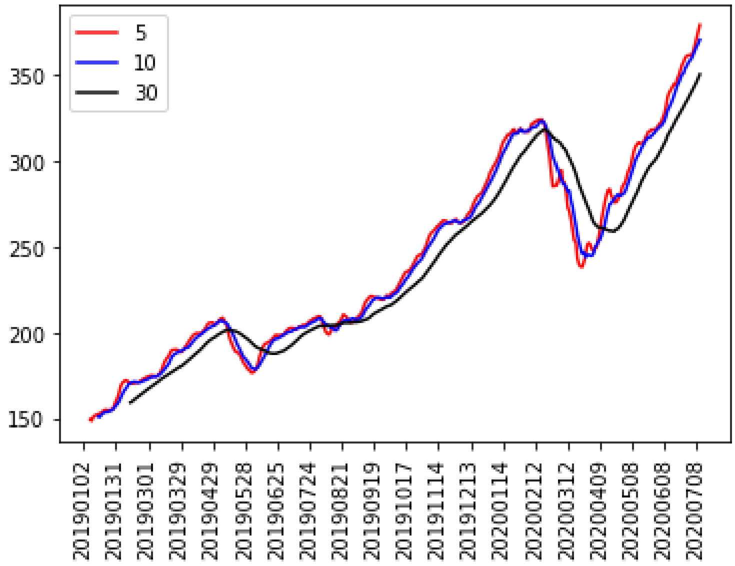
请下载苹果公司（AAPL）的股票价格（从2019年初至今），试求

6.1

画出其收盘价的时间序列图及五日均线、十日均线、三十日均线；
代码：

```
import tushare as ts
import matplotlib.pyplot as plt
from matplotlib.pylab import date2num
import pandas as pd
import numpy as np

pro = ts.pro_api()

df = pro.us_daily(ts_code='AAPL', start_date='20190101')
df = df.sort_values(by='trade_date', ascending=True)
df['trade_date2'] = df['trade_date'].copy()
df['trade_date'] = pd.to_datetime(df['trade_date']).map(date2num)
df['dates'] = np.arange(0, len(df))

for ma in [5,10,30]:
    df[str(ma)] = df.close.rolling(ma).mean()

x= df.dates
x_ticks_label = df.trade_date2

plt.plot(df[['dates']],df['5'],color='red',label='5')
plt.plot(df[['dates']],df['10'],color='blue',label='10')
plt.plot(df[['dates']],df['30'],color='black',label='30')
plt.xticks(x[::20],x_ticks_label[::20],rotation=90)
plt.legend()
plt.show()
```

运行：

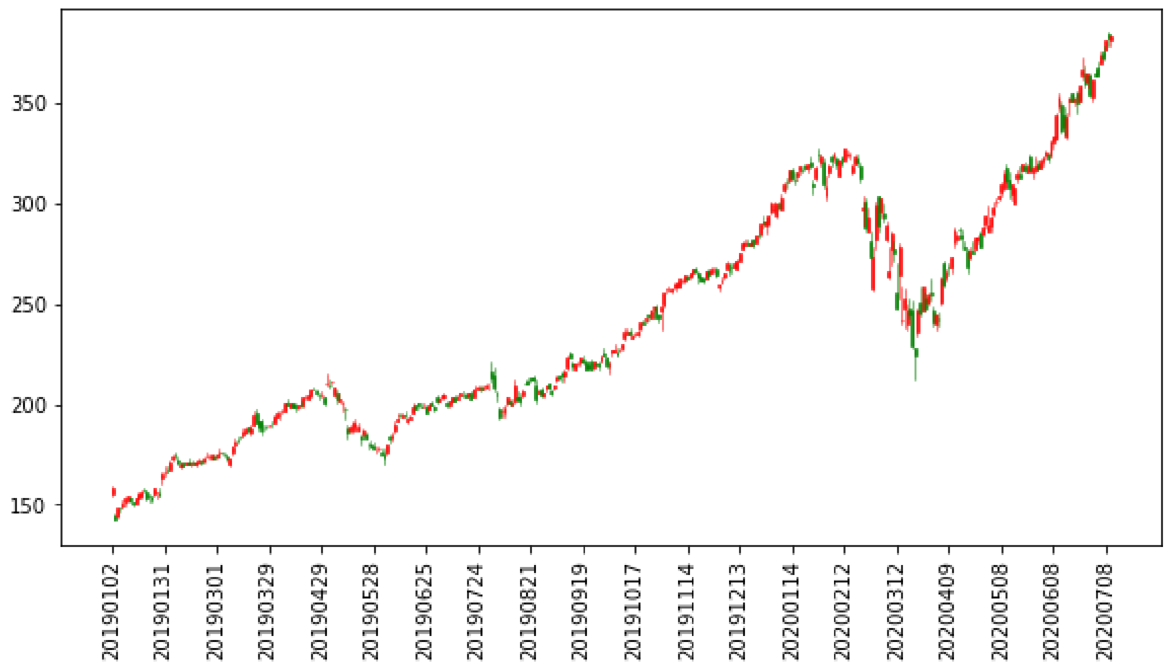


6.2

试对AAPL股票价画其烛线图（股票条形图）
代码：

```
import tushare as ts
import matplotlib.pyplot as plt
from matplotlib.pylab import date2num
import pandas as pd
import numpy as np
import mpl_finance

pro = ts.pro_api()

df = pro.us_daily(ts_code='AAPL', start_date='20190101')
df = df.sort_values(by='trade_date', ascending=True)
df['trade_date2'] = df['trade_date'].copy()
df['trade_date'] = pd.to_datetime(df['trade_date']).map(date2num)
df['dates'] = np.arange(0, len(df))

fig, ax = plt.subplots(figsize=(10,5))

mpl_finance.candlestick_ochl(
    ax=ax,
    quotes=df[['dates', 'open', 'close', 'high', 'low']].values,
    width=0.7,
    colorup='r',
    colordown='g',
    alpha=0.7)

x= df.dates
x_ticks_label = df.trade_date2
plt.xticks(x[::20],x_ticks_label[::20],rotation=90)
plt.show()
```

运行：

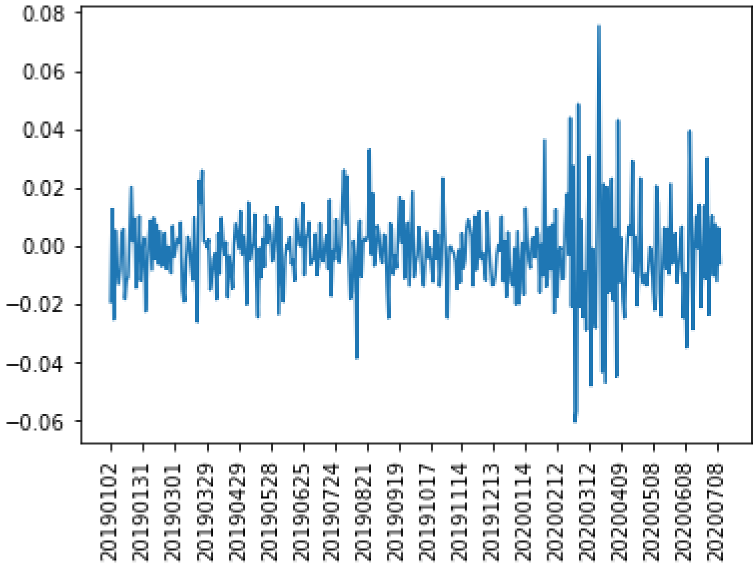


6.3

计算收盘价的对数收益率，并画出其收益率图
代码：

```
import tushare as ts
import math
import matplotlib.pyplot as plt

pro = ts.pro_api()

#获取单一股票行情
df = pro.us_daily(ts_code='AAPL', start_date='20190101')
df = df.sort_values(by='trade_date', ascending=True)

df['rate_log'] = df.apply(lambda x: math.log((x['open'] / x['close']),math.e),
axis=1)

x = df.trade_date
y = df.rate_log

plt.plot(x,y)
plt.xticks(x[::20],rotation=90)
plt.show()
```

运行：



6.4

求出其ACF图（lag=24）
代码：

```
import tushare as ts
import math
from statsmodels.graphics.tsaplots import plot_acf

pro = ts.pro_api()

#获取单一股票行情
df = pro.us_daily(ts_code='AAPL', start_date='20190101')
df = df.sort_values(by='trade_date', ascending=True)

df['rate_log'] = df.apply(lambda x: math.log((x['open'] / x['close']),math.e),
axis=1)

plot_acf(df['rate_log'])
```

运行：