Python 数据分析实践（Data Analysis Action）

Chap 5 网络数据收集和分析 Web Data Collection and Analysis

内容：

- 数据分析实战：本地数据和Web数据

实践：

- 数据统计性描述
- 可视化绘图
- 网络数据采集和分析
  - 新闻网页数据采集
  - 股票金融数据分析
- BeautifulSoup4
- requests

实例：

- 实例1：本地数据访问和分析（csv和时间序列）
- 实例2：新闻网页数据采集
- 实例3：根据API获取股票金融数据

本节课讲述了从本地和Web获取数据，并进行数据的预处理、分析、可视化和磁盘保存。

两个重要的Python库处理网络资源：

- BeautifulSoup4

  > BeautifulSoup4是爬虫必学的技能，最主要的功能是从网页抓取数据，进行HTML/XML的解析。Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。

- requests

  > requests是一个重要的Python第三方库，访问和处理URL资源特别方便（Python内置的urllib模块使用比较麻烦，而且缺少很多实用的高级功能）

本章目录：

1. 读入本地磁盘数据，并进行数据分析、统计性描述、可视化绘图
2. 采集网络新闻标题数据，解析新闻网页内容
3. 获取Web数据（股票数据），并进行股票数据的分析和处理，存入本地磁盘

In [1]:
```
# 必要准备工作：导入库，配置环境等
#from __future__ import division
#import os, sys

# 导入库并为库起个别名
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt
```

实例1：本地数据访问和分析（csv和时间序列）

本地数据集1：餐馆小费

tips.csv 是个关于餐馆小费记录的数据，包含七个字段（total_bill，tip，sex，smoker，day，time，size），共计244条记录。

- 磁盘读入csv格式文件转为pd数据结构
- 对数据分析（缺失值-填充，清理，汇总描述，可视化绘图）
- 数据相关性分析
- 数据分组聚合

In [2]:
```
import pandas as pd
tips = 'data/tips.csv'
data = pd.read_csv(tips)  # 默认header='infer'，推导第一行是header，小费记录始于第二行
print(len(data)) # 数据记录数量
data.head()  # 预览最前5行记录
#data.tail()  # 预览最后5行记录
```

```
244
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [3]:
```
data.describe() # 数据的汇总描述
```

Out[3]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [4]:
```
stat = data.describe() # 数据的基本统计量
# 重点：可以自己添加统计量信息
stat.loc['range'] = stat.loc['max'] - stat.loc['min']  # 极差 range
stat.loc['var'] = stat.loc['std'] / stat.loc['mean']  # 变异系数，标准差/均值的离中趋势
stat.loc['IQR'] =  stat.loc['75%'] - stat.loc['25%'] # 四分位数间距（极差）
stat
```

Out[4]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000
range	47.740000	9.000000	5.000000
var	0.449936	0.461478	0.370125
IQR	10.780000	1.562500	1.000000

In [5]:
```
# 数据的其他统计量
# 数据的分布统计
print(data.median())  # 数据的中位数
data.mode() # 数据的众数
data.quantile(0.1)  # 数据的百分位数 data.quantile(q=0.5)

# 数据的离中趋势度量
data.skew() # 数据的偏度
data.kurt() # 数据的峰度

# 数据列之间的相关度量
#协方差确定两个变量的关系，即正相关，负相关/无关
#相关系数确定两个变量的关系&相关程度
data.cov() # 数据的协方差矩阵,只表示线性相关的方向，取值正无穷到负无穷,协方差为正值说明正相关,负值
data.corr() # 数据的Pearson相关系数矩阵
```

```
total_bill    17.795
tip            2.900
size           2.000
dtype: float64
```

Out[5]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [6]:
```
data['tip'].cov(data['size'])  #计算指定变量之间的协方差
```

Out[6]: 0.6439064291978684

In [7]:
```
data['tip'].corr(data['size'])  #计算指定变量之间的相关系数
```

Out[7]: 0.4892987752303577

In [8]:
```
data.columns
```

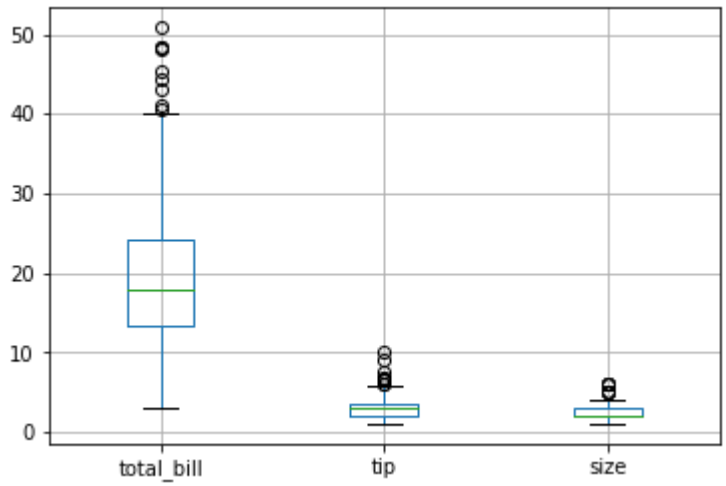
Out[8]: Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')

In [9]:
```
data.columns.names  # 此时每个列没有name
```

Out[9]: FrozenList([None])

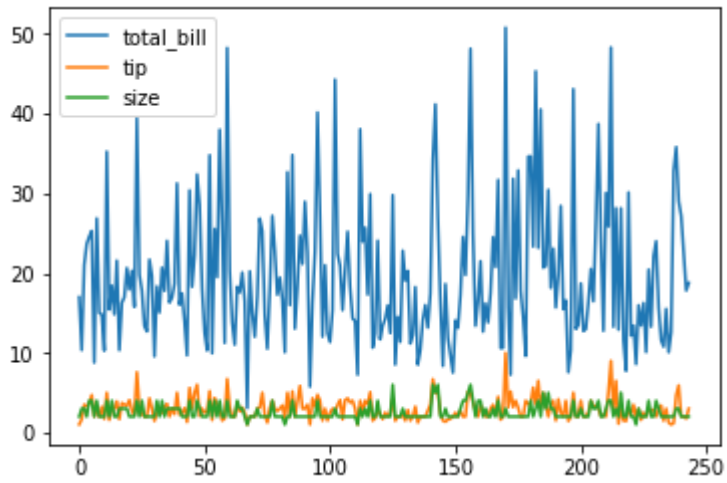
In [10]:
```
# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt

data.boxplot(return_type='axes')  # 画盒图，直接使用DataFrame的方法
#data.boxplot()  # 画盒图，直接使用DataFrame的方法，需要屏蔽warning
#data[['tip','size']].boxplot(return_type='axes') # 只对两个列画盒图
```

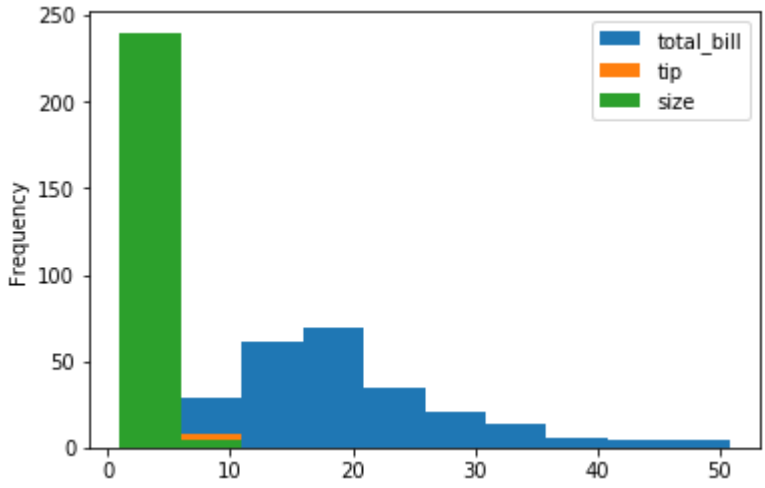
Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f951992b150>
```



In [11]:
```
data.plot.line()  # 线图
data.plot.hist()  # 直方图
```

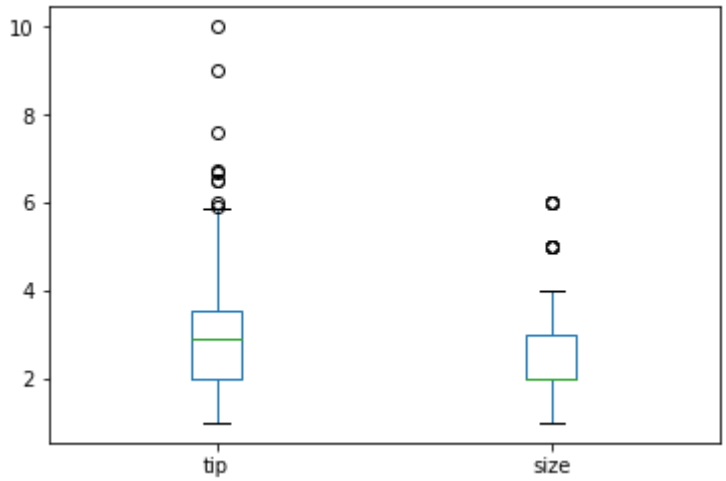
Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9512b8d050>
```





In [12]:
```
# 几种盒图绘图
#data[['tip','size']].boxplot() # 盒图1，同前面
#data[['tip','size']].plot(kind='box')  # 盒图2
data[['tip','size']].plot.box()  # 盒图3
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9512b00a10>



In [13]:
```
# 其他多种图类型, tip, size, total_bill
#data['tip'].plot.line()  # 线图 1
#data['tip'].plot(kind='line')  # 线图 2
#data['tip'].plot.hist()  # 直方图
#data['tip'].plot.kde()  # 密度图1 'kde' : Kernel Density Estimation plot
#data['tip'].plot.density() # 密度图1 density,同上
#data['tip'].plot.pie() # 饼图
```

In [14]:
```
# 相关性分析
data.corr()  # method : {'pearson', 'kendall', 'spearman'}, 默认pearson
```

Out[14]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [15]:
```
data.corr(method='kendall')  # method : {'pearson', 'kendall', 'spearman'}
```

Out[15]:

	total_bill	tip	size
total_bill	1.000000	0.517181	0.484342
tip	0.517181	1.000000	0.378185
size	0.484342	0.378185	1.000000

In [16]:
```
data.head()
```

Out[16]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

思考： 数据分组进一步考虑：消费与date（周一周末）是否有关？是否与time（中餐晚餐）有关？

怎么做?

- 把day和time两个列转为行索引的外层和内层
- DataFrame的set_index函数会将其一个或多个列转换为行索引，并创建一个新的DataFrame。

In [17]:
```
# 考虑bill与date（周一周末）是否有关？是否与time（中餐晚餐）有关？
# 把day和time两个列转为行索引的外层和内层
# DataFrame的set_index函数会将其一个或多个列转换为行索引，并创建一个新的DataFrame。
data2 = data.set_index(['day', 'time'])
data2.head()
```

Out[17]:

		total_bill	tip	sex	smoker	size
day	**time**					
Sun	**Dinner**	16.99	1.01	Female	No	2
	Dinner	10.34	1.66	Male	No	3
	Dinner	21.01	3.50	Male	No	3
	Dinner	23.68	3.31	Male	No	2
	Dinner	24.59	3.61	Female	No	4

In [18]:
```
data2.tail()
```

Out[18]:

		total_bill	tip	sex	smoker	size
day	**time**					
Sat	**Dinner**	29.03	5.92	Male	No	3
	Dinner	27.18	2.00	Female	Yes	2
	Dinner	22.67	2.00	Male	Yes	2
	Dinner	17.82	1.75	Male	No	2
Thur	**Dinner**	18.78	3.00	Female	No	2

In [19]:
```
# 选取周日Sun的消费统计汇总情况
data2.loc['Sun'].describe()
```

Out[19]:

	total_bill	tip	size
count	76.000000	76.000000	76.000000
mean	21.410000	3.255132	2.842105
std	8.832122	1.234880	1.007341
min	7.250000	1.010000	2.000000
25%	14.987500	2.037500	2.000000
50%	19.630000	3.150000	2.000000
75%	25.597500	4.000000	4.000000
max	48.170000	6.500000	6.000000

In [20]:
```
# 选取周日Sun晚餐Dinner的消费统计汇总情况
data2.loc['Sun'].loc['Dinner'].describe()
#data2.ix['Sun'].ix['Lunch'].describe()  # 周日没有Lunch消费的记录
```

Out[20]:

	total_bill	tip	size
count	76.000000	76.000000	76.000000
mean	21.410000	3.255132	2.842105
std	8.832122	1.234880	1.007341
min	7.250000	1.010000	2.000000
25%	14.987500	2.037500	2.000000
50%	19.630000	3.150000	2.000000
75%	25.597500	4.000000	4.000000
max	48.170000	6.500000	6.000000

In [21]:
```
# 对比工作日周五的午餐和晚餐消费均值
print(data2.loc['Fri'].loc['Lunch'].mean()) # 选取周五Fri午餐Lunch的消费统计汇总情况
print(data2.loc['Fri'].loc['Dinner'].mean()) # 选取周五Fri晚餐Dinner的消费统计汇总情况
```

```
total_bill    12.845714
tip            2.382857
size           2.000000
dtype: float64
total_bill    19.663333
tip            2.940000
size           2.166667
dtype: float64
```

In [22]:
```
# 对比周五到周日的消费均值
print(data2.loc['Fri']['total_bill'].mean())
print(data2.loc['Sat']['total_bill'].mean())
print(data2.loc['Sun']['total_bill'].mean())
```

```
17.151578947368417
20.441379310344825
21.410000000000004
```

In [23]:
```
# 交换索引（行索引的内层和外层索引交换）
data3 = data2.swaplevel(0,1)  # 交换索引后返回新的data3
data3.tail()
```

Out[23]:

		total_bill	tip	sex	smoker	size
time	**day**					
Dinner	**Sat**	29.03	5.92	Male	No	3
	Sat	27.18	2.00	Female	Yes	2
	Sat	22.67	2.00	Male	Yes	2
	Sat	17.82	1.75	Male	No	2
	Thur	18.78	3.00	Female	No	2

In [24]:
```
# 比较午餐Lunch和晚餐Dinner的消费统计汇总情况
print(data3.loc['Dinner'].describe())
print(data3.loc['Lunch'].describe())
```

```
       total_bill         tip        size
count  176.000000  176.000000  176.000000
mean    20.797159    3.102670    2.630682
std      9.142029    1.436243    0.910241
min      3.070000    1.000000    1.000000
25%     14.437500    2.000000    2.000000
50%     18.390000    3.000000    2.000000
75%     25.282500    3.687500    3.000000
max     50.810000   10.000000    6.000000
       total_bill        tip       size
count   68.000000  68.000000  68.000000
mean    17.168676   2.728088   2.411765
std      7.713882   1.205345   1.040024
min      7.510000   1.250000   1.000000
25%     12.235000   2.000000   2.000000
50%     15.965000   2.250000   2.000000
75%     19.532500   3.287500   2.000000
max     43.110000   6.700000   6.000000
```

其实，我们不必进行上面的操作，因为pandas提供了非常方便的groupby分组操作

groupby分组操作

pandas的DataFrame有groupby操作，可以非常方便对数据分组。不需要将多个列索引转换为行索引的情况下，可以直接对数据进行分组分析计算。

- df.groupby(['col2', 'col3']) # 首先，按照col2和col3的不同值进行分组
- df['col1'].describe() # 然后，统计col1的汇总情况

In [25]:
```
# 添加“小费占总额百分比”列
data['tip_pct'] = data['tip'] / data['total_bill']
data[:6]
```

Out[25]:

	total_bill	tip	sex	smoker	day	time	size	tip_pct
0	16.99	1.01	Female	No	Sun	Dinner	2	0.059447
1	10.34	1.66	Male	No	Sun	Dinner	3	0.160542
2	21.01	3.50	Male	No	Sun	Dinner	3	0.166587
3	23.68	3.31	Male	No	Sun	Dinner	2	0.139780
4	24.59	3.61	Female	No	Sun	Dinner	4	0.146808
5	25.29	4.71	Male	No	Sun	Dinner	4	0.186240

In [26]:
```
# 分组统计
data.groupby(['sex','smoker']).count() # 统计不同性别和是否抽烟的数量
```

Out[26]:

		total_bill	tip	day	time	size	tip_pct
sex	**smoker**						
Female	**No**	54	54	54	54	54	54
	Yes	33	33	33	33	33	33
Male	**No**	97	97	97	97	97	97
	Yes	60	60	60	60	60	60

In [27]:
```
# 分组统计
data.groupby(['day','time']).count()  # 统计不同天和不同餐时的数量
```

Out[27]:

		total_bill	tip	sex	smoker	size	tip_pct
day	**time**						
Fri	**Dinner**	12	12	12	12	12	12
	Lunch	7	7	7	7	7	7
Sat	**Dinner**	87	87	87	87	87	87
Sun	**Dinner**	76	76	76	76	76	76
Thur	**Dinner**	1	1	1	1	1	1
	Lunch	61	61	61	61	61	61

In [28]:
```
# 统计不同天和时间的平均情况
data.groupby(['day','time']).mean()
```

Out[28]:

day	time	total_bill	tip	size	tip_pct
Fri	Dinner	19.663333	2.940000	2.166667	0.158916
	Lunch	12.845714	2.382857	2.000000	0.188765
Sat	Dinner	20.441379	2.993103	2.517241	0.153152
Sun	Dinner	21.410000	3.255132	2.842105	0.166897
Thur	Dinner	18.780000	3.000000	2.000000	0.159744
	Lunch	17.664754	2.767705	2.459016	0.161301

In [29]:
```
# 只统计不同天和时间的tip平均情况
data['tip'].groupby([data['day'], data['time']]).mean()
```

Out[29]:
```
day   time
Fri   Dinner    2.940000
      Lunch     2.382857
Sat   Dinner    2.993103
Sun   Dinner    3.255132
Thur  Dinner    3.000000
      Lunch     2.767705
Name: tip, dtype: float64
```

In [30]:
```
# 比较不同性别在不同天的午餐Lunch和晚餐Dinner的平均小费情况
data['tip'].groupby([data['sex'], data['time']]).mean()
```

Out[30]:
```
sex     time
Female  Dinner    3.002115
        Lunch     2.582857
Male    Dinner    3.144839
        Lunch     2.882121
Name: tip, dtype: float64
```

In [31]:
```
# 综合比较不同性别在周末午餐Lunch和晚餐Dinner的平均消费情况
data.groupby(['sex', 'time']).mean()
```

Out[31]:

sex	time	total_bill	tip	size	tip_pct
Female	Dinner	19.213077	3.002115	2.461538	0.169322
	Lunch	16.339143	2.582857	2.457143	0.162285
Male	Dinner	21.461452	3.144839	2.701613	0.155407
	Lunch	18.048485	2.882121	2.363636	0.166083

In [32]:
```
# 分组统计不同性别给出小费的比例情况
grouped = data.groupby(['sex'])
grouped.mean()
```

Out[32]:

sex	total_bill	tip	size	tip_pct
Female	18.056897	2.833448	2.459770	0.166491
Male	20.744076	3.089618	2.630573	0.157651

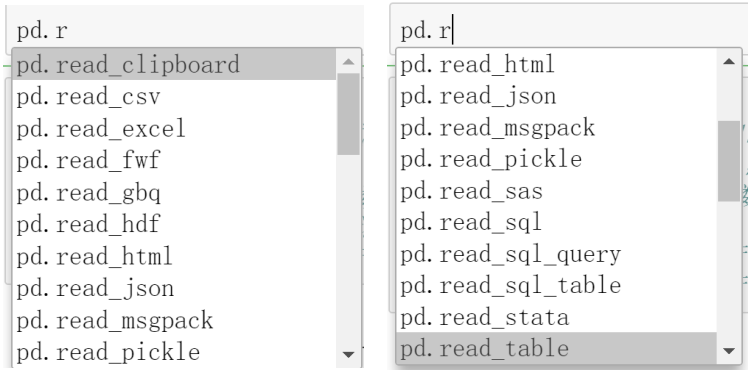
In [33]:
```
# 不同性别给小费比例的均值
grouped['tip_pct'].mean()
```

Out[33]:
```
sex
Female    0.166491
Male      0.157651
Name: tip_pct, dtype: float64
```

In [34]:
```
# 首先，根据sex和smoker对tips进行分组
grouped = data['tip_pct'].groupby([data['sex'],data['smoker']])
grouped.mean()
```

Out[34]:
```
sex     smoker
Female  No        0.156921
        Yes       0.182150
Male    No        0.160669
        Yes       0.152771
Name: tip_pct, dtype: float64
```

pandas可以读入的数据文件格式包括：



pandas读入多种数据格式：
- csv, excel
- html, json
- pickle
- 剪贴板
- 数据库, sql, hdf
- SASXport，Google BigQuery等
- 通用表格, table

本地数据集2：股票时间序列数据

stock_px.csv 是个关于股票价格的时间序列数据，包含9只股票，对应9个字段（AA，AAPL，GE，IBM，JNJ，MSFT，PEP，SPX，XOM），时间从1990/2/1到2011/10/14，共计5472条记录。

美铝公司[AA]，苹果公司[AAPL]，通用电气[GE]，微软[MSFT]，强生[JNJ]，百事[PEP]，美国标准普尔500指数 (SPX)，埃克森美孚[XOM]

In [35]:
```
import pandas as pd
import numpy as np
from datetime import datetime

f = 'data/stock_px.csv'
data = pd.read_csv(f, index_col='date')  # 使用date列作为行索引
data.index = pd.to_datetime(data.index)  # 将字符串索引转换成时间索引
print(len(data)) # 数据记录数量
data.head()  # 预览最前5行记录
data.tail()  # 预览最后5行记录
```

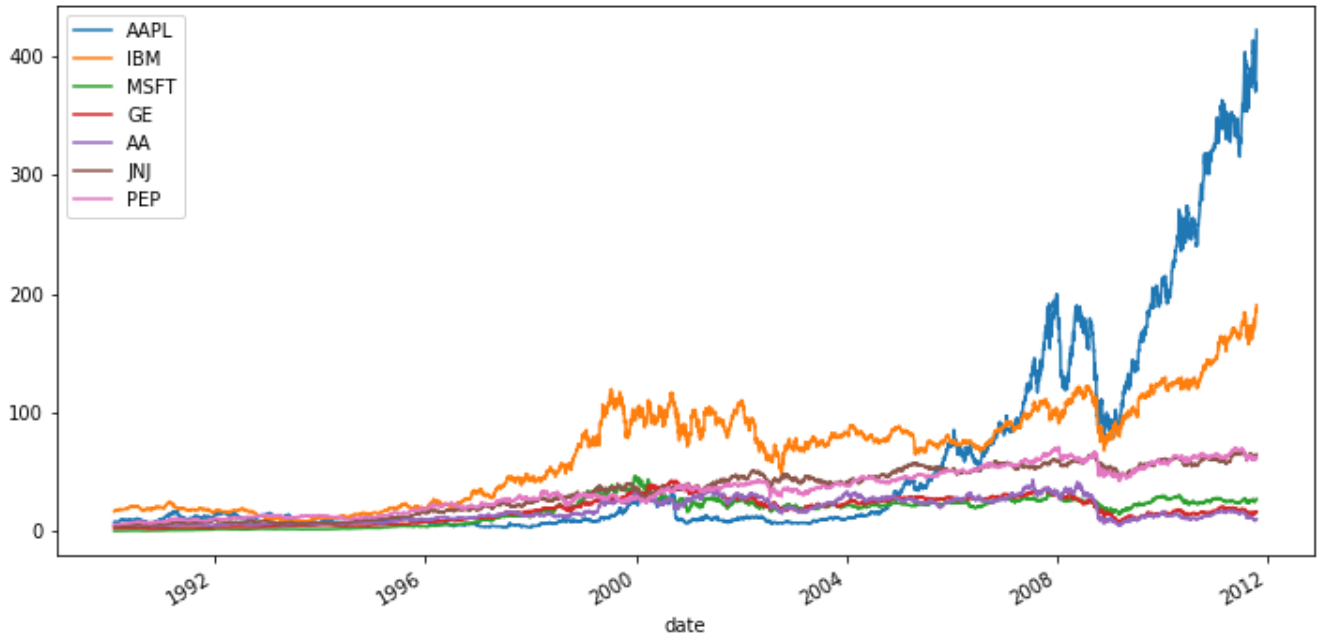
```
5472
```

Out[35]:

date	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
2011-10-10	10.09	388.81	16.14	186.62	64.43	26.94	61.87	1194.89	76.28
2011-10-11	10.30	400.29	16.14	185.00	63.96	27.00	60.95	1195.54	76.27
2011-10-12	10.05	402.19	16.40	186.12	64.33	26.96	62.70	1207.25	77.16
2011-10-13	10.10	408.43	16.22	186.82	64.23	27.18	62.36	1203.66	76.37
2011-10-14	10.26	422.00	16.60	190.53	64.72	27.27	62.24	1224.58	78.11

In [36]:
```
plt.rc('figure', figsize=(12,6))
data[['AAPL','IBM','MSFT','GE','AA','JNJ','PEP']].plot.line() # 绘曲线图
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9510761a10>
```



In [37]:
```
# 重点了解苹果股票的基本统计量
data['AAPL'].describe()
```

Out[37]:
```
count    5472.000000
mean       57.119313
std        88.670423
min         3.230000
25%         8.760000
50%        11.990000
75%        68.017500
max       422.000000
Name: AAPL, dtype: float64
```

In [38]:
```
# 苹果股票的其他统计量
print(data['AAPL'].median())  # 数据的中位数
print(data['AAPL'].mode()) # 数据的众数
print(data['AAPL'].quantile(0.1))  # 数据的百分位数 data.quantile(q=0.5)
#print(data['AAPL'].skew()) # 数据的偏度
#print(data['AAPL'].kurt()) # 数据的峰度
#data.mode?  # 了解mode用法
#data.quantile?  # 了解quantile用法
```

```
11.99
0    9.28
dtype: float64
6.51
```

In [39]:
```
# 各股票之间的相关统计量
#data.cov() # 数据的协方差矩阵
data.corr() # 数据的Pearson相关系数矩阵
```

Out[39]:

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
AA	1.000000	0.101313	0.916804	0.600211	0.685752	0.776796	0.634679	0.846861	0.567025
AAPL	0.101313	1.000000	0.142381	0.749037	0.651564	0.423274	0.741942	0.410813	0.781369
GE	0.916804	0.142381	1.000000	0.681659	0.717824	0.875398	0.652904	0.935598	0.581171
IBM	0.600211	0.749037	0.681659	1.000000	0.902894	0.871615	0.885029	0.835484	0.855195
JNJ	0.685752	0.651564	0.717824	0.902894	1.000000	0.846906	0.970478	0.845401	0.925600
MSFT	0.776796	0.423274	0.875398	0.871615	0.846906	1.000000	0.781791	0.949715	0.730557
PEP	0.634679	0.741942	0.652904	0.885029	0.970478	0.781791	1.000000	0.816477	0.964278
SPX	0.846861	0.410813	0.935598	0.835484	0.845401	0.949715	0.816477	1.000000	0.761077
XOM	0.567025	0.781369	0.581171	0.855195	0.925600	0.730557	0.964278	0.761077	1.000000

In [40]:
```
data.head()
```

Out[40]:

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
date									
1990-02-01	4.98	7.86	2.87	16.79	4.27	0.51	6.04	328.79	6.12
1990-02-02	5.04	8.00	2.87	16.89	4.37	0.51	6.09	330.92	6.24
1990-02-05	5.07	8.18	2.87	17.32	4.34	0.51	6.05	331.85	6.25
1990-02-06	5.01	8.12	2.88	17.56	4.32	0.51	6.15	329.66	6.23
1990-02-07	5.04	7.77	2.91	17.93	4.38	0.51	6.17	333.75	6.33

In [41]:
```
data.tail()
```

Out[41]:

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
date									
2011-10-10	10.09	388.81	16.14	186.62	64.43	26.94	61.87	1194.89	76.28
2011-10-11	10.30	400.29	16.14	185.00	63.96	27.00	60.95	1195.54	76.27
2011-10-12	10.05	402.19	16.40	186.12	64.33	26.96	62.70	1207.25	77.16
2011-10-13	10.10	408.43	16.22	186.82	64.23	27.18	62.36	1203.66	76.37
2011-10-14	10.26	422.00	16.60	190.53	64.72	27.27	62.24	1224.58	78.11

股票的时间序列数据分析

除了前面常用的统计、汇总、分组、可视化分析，对于股票数据，还可以进行更复杂的数据分析任务：

- 根据每天的收盘价返回对数收益率

In [42]:
```
# 只取出苹果股票分析
df = pd.DataFrame(data['AAPL'],index=data.index, columns=['AAPL'])  # data['AAPL']只是个Series
df.tail()
```

Out[42]:

date	AAPL
2011-10-10	388.81
2011-10-11	400.29
2011-10-12	402.19
2011-10-13	408.43
2011-10-14	422.00

In [43]:
```
# 更复杂的数据分析任务：根据每天的收盘价返回对数收益率
# 首先添加包含对应信息的列，生成一个新的列，
# 然后中所有股价上进行循环，逐步计算单个对数收益率值
df['Return'] = 0.0
for i in range(1, len(df)):
    df['Return'][i] = np.log(df['AAPL'][i] / df['AAPL'][i-1])
df.tail()
```

Out[43]:

date	AAPL	Return
2011-10-10	388.81	0.050128
2011-10-11	400.29	0.029098
2011-10-12	402.19	0.004735
2011-10-13	408.43	0.015396
2011-10-14	422.00	0.032685

In [44]:
```
# 也可以使用向量化代码，在不使用循环的情况下得到相同的结果，即shift方法
df['Return2'] = np.log(df['AAPL'] / df['AAPL'].shift(1))
df[['AAPL', 'Return', 'Return2']].tail()
# 最后面两列的值相同：更紧凑和更容易理解的代码，而且是更快速的替代方案
```

Out[44]:

date	AAPL	Return	Return2
2011-10-10	388.81	0.050128	0.050128
2011-10-11	400.29	0.029098	0.029098
2011-10-12	402.19	0.004735	0.004735
2011-10-13	408.43	0.015396	0.015396
2011-10-14	422.00	0.032685	0.032685

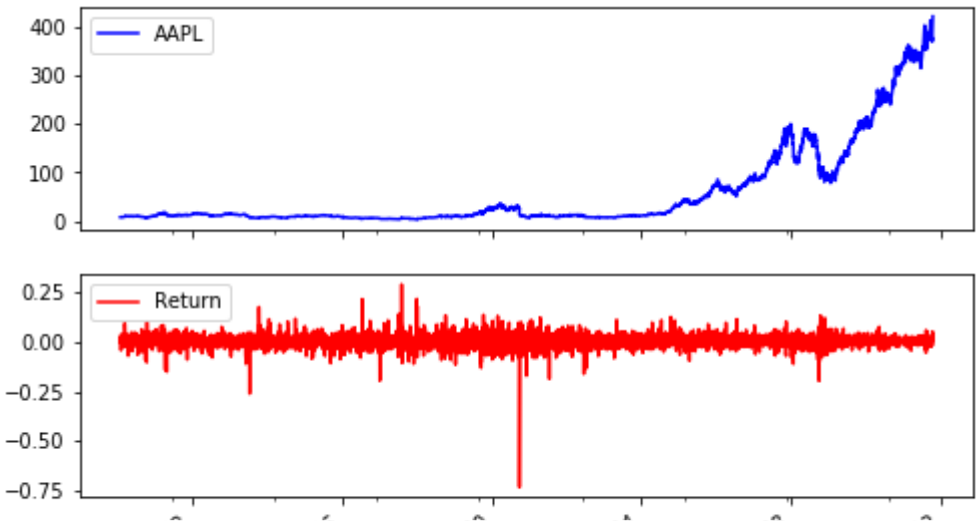
In [45]:
```
# 目前，一个对数收益率数据列就足够了，可以删除另一个列
del df['Return2'] # 删除列
df.tail()
```

Out[45]:

date	AAPL	Return
2011-10-10	388.81	0.050128
2011-10-11	400.29	0.029098
2011-10-12	402.19	0.004735
2011-10-13	408.43	0.015396
2011-10-14	422.00	0.032685

In [46]:
```
# 绘图更好地概览股价和波动率变化
df[['AAPL', 'Return']].plot(subplots=True, style=['b','r'], figsize=(8,5))
```

Out[46]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f95107b6a90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f951061fd90>],
      dtype=object)
```



In [47]:
```
# 技术型股票交易者可能对移动平均值（即趋势）更感兴趣，
# 移动平均值很容易使用pandas的rolling_mean计算
#df['42d'] = pd.rolling_mean(df['AAPL'], window = 42) # 过时，现在不用
df['42d'] = df['AAPL'].rolling(window=42,center=False).mean()
#df['252d'] = pd.rolling_mean(df['AAPL'], window = 252) # 过时，现在不用
df['252d'] = df['AAPL'].rolling(window = 252,center=False).mean()
df[['AAPL', '42d', '252d']].tail()
```

Out[47]:

date	AAPL	42d	252d
2011-10-10	388.81	384.502381	346.165278
2011-10-11	400.29	385.135476	346.569048
2011-10-12	402.19	385.735476	346.974008
2011-10-13	408.43	386.331190	347.395119
2011-10-14	422.00	387.319762	347.820754

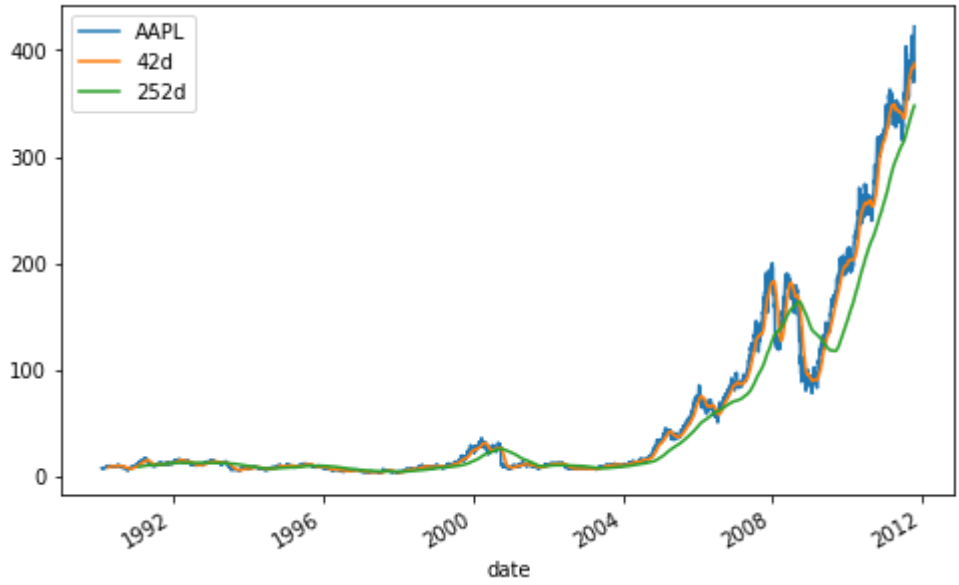
In [48]:
```
df[['AAPL', '42d', '252d']].head() # 对于后两列，前面的数据为空
```

Out[48]:

date	AAPL	42d	252d
1990-02-01	7.86	NaN	NaN
1990-02-02	8.00	NaN	NaN
1990-02-05	8.18	NaN	NaN
1990-02-06	8.12	NaN	NaN
1990-02-07	7.77	NaN	NaN

In [49]:
```
# 包含两种趋势的典型股价图表绘图
df[['AAPL', '42d', '252d']].plot(figsize=(8,5))
```

Out[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9510593fd0>
```

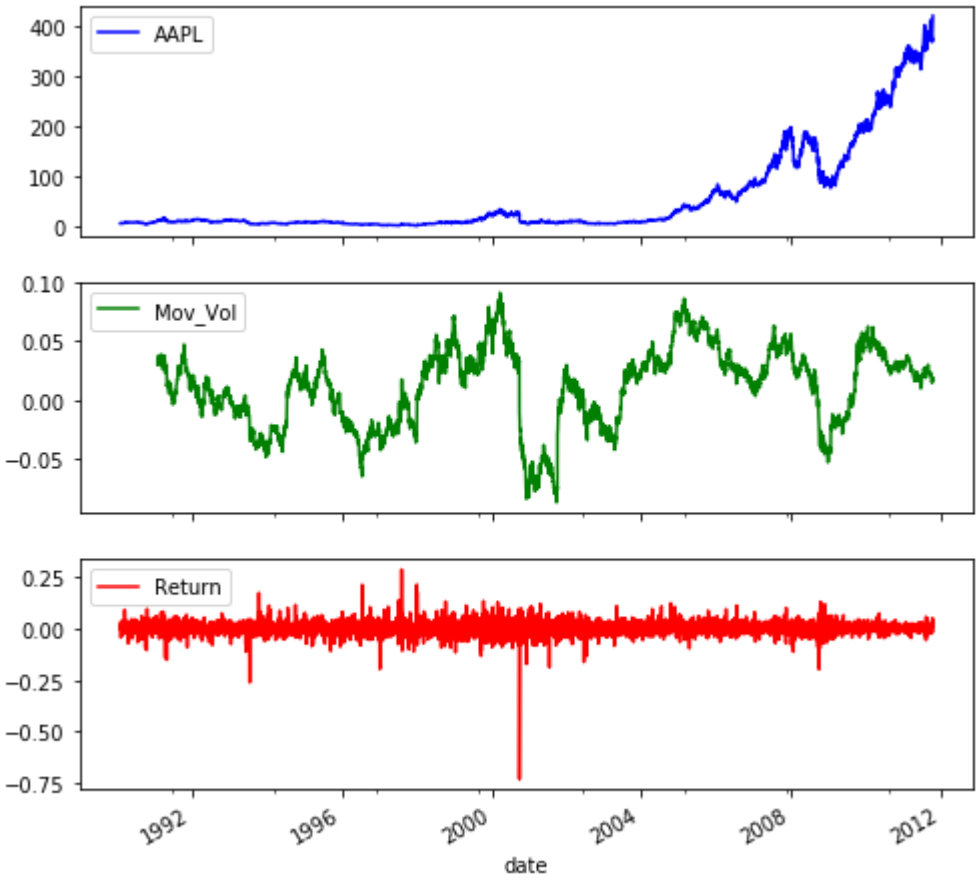


In [50]:
```
# 期权交易者更喜欢的话题，对数收益率的移动历史标准差--即移动历史波动率
import math
df['Mov_Vol'] = df['Return'].rolling(window=252,center=False).mean() * math.sqrt(252)
df['Mov_Vol'].tail() # Mov_Vol列的数据在前面252行为空
```

Out[50]:
```
date
2011-10-10    0.017317
2011-10-11    0.018475
2011-10-12    0.018437
2011-10-13    0.018953
2011-10-14    0.018474
Name: Mov_Vol, dtype: float64
```

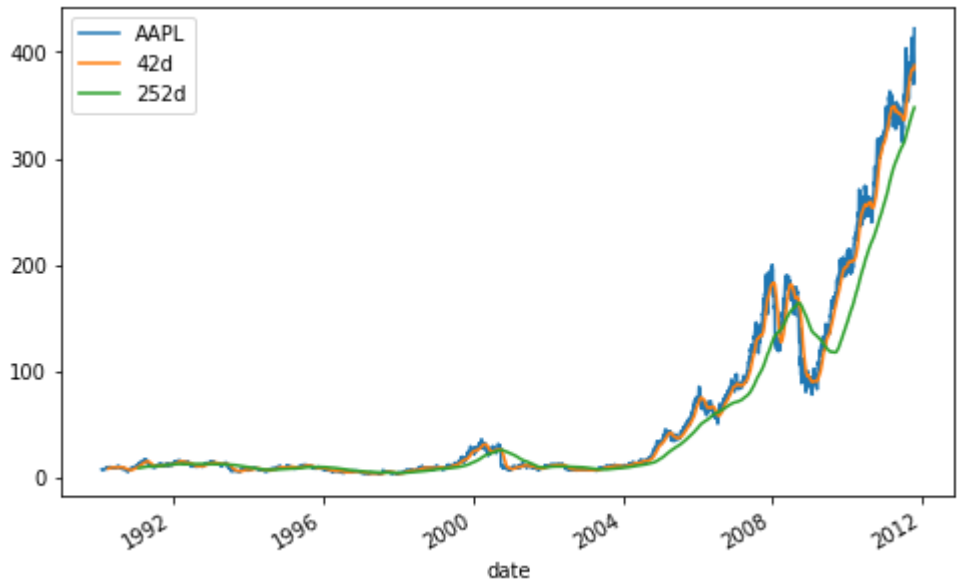
In [51]:
```
# 杠杆效应假设，说明市场下跌时历史移动波动率倾向于升高，而在市场上涨时波动率下降
df[['AAPL', 'Mov_Vol', 'Return']].plot(subplots=True, style=['b','g','r'], figsize=(8,8))
```

Out[51]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f9512bc5710>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f95104f7d90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f95104a5c10>],
      dtype=object)
```



In [52]:
```
# 包含两种趋势的典型股价图表绘图
df[['AAPL', '42d', '252d']].plot(figsize=(8,5))
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7f95103adb10>



In [53]:
```
df.tail()
```

Out[53]:

date	AAPL	Return	42d	252d	Mov_Vol
2011-10-10	388.81	0.050128	384.502381	346.165278	0.017317
2011-10-11	400.29	0.029098	385.135476	346.569048	0.018475
2011-10-12	402.19	0.004735	385.735476	346.974008	0.018437
2011-10-13	408.43	0.015396	386.331190	347.395119	0.018953
2011-10-14	422.00	0.032685	387.319762	347.820754	0.018474

保存数据

现在我们想把对苹果股票的分析数据保存下来，在以后的分析中继续使用。pandas的DataFrame的保存数据类型，参考前面表格中的读入数据类型。

In [54]:
```
# 为了以后更容易导入数据，我们生成一个新的csv数据文本，并将所有数据行写入新文件
out_file = open('data/aapl.csv', 'w')
df.to_csv(out_file)
out_file.close()
```

实例2：新闻网页数据采集

以中国新闻网 (http://www.chinanews.com/ (http://www.chinanews.com/)) 为例。

本课件分为如下两个部分：

- **1 单个页面新闻标题爬取**: 主要介绍如何从一个页面中利用 BeautifulSoup 寻找感兴趣的内容
- **2 多页面新闻标题爬取**
  - 2.1 滚动新闻页面爬取: 通过有规律的url进行爬取
  - 2.2 获取各板块头条新闻以及热点新闻： 自动获取不同板块地址进行内容爬取

In [55]:
```
from IPython.display import IFrame
url="http://www.chinanews.com/"
IFrame(url, width='100%', height=400)
```

Out[55]:

In [56]:
```
# 加载访问url资源的库
from bs4 import BeautifulSoup
import requests
```

In [57]:
```
url = "http://www.chinanews.com/"
html = requests.get(url)                              # 获取网页
soup = BeautifulSoup(html.content,"html.parser")  # 将该网页转化为 BeautifulSoup 对象
#print(soup.prettify())                              # 格式化打印网页，确认获取数据
```

1 单页面新闻标题爬取

大标题新闻的爬取

打开中国新闻网，我们首先对主页上最主要的新闻进行爬取，如下图所示，这些加粗部分是着重强调的新闻，我们首先获取这部分数据

通过浏览器的开发者模式（F12）,我们可以定位到该部分内容在一个div框架下，如何找到这个div呢？其有唯一属性值：class = xwzxdd-dbt

In [58]:
```
# 大标题新闻
bignews = soup.findAll(name="div", attrs={"class" :"xwzxdd-dbt"})  # 通过class="xwzxdd-dbt"查找到

for b in bignews:
    for title in b.findAll(name="a"):
#         print(title)
#         <a href="//www.chinanews.com/gn/2020/03-22/9133889.shtml">交通运输部：全面纠正硬隔离等不
#         <a class="ptv" href="//www.chinanews.com/shipin/spfts/20200321/2659.shtml">回看</a>
        if len(title.attrs)==1:  # 排除附加类视频新闻，附加视频类新闻有class="ptv"的属性（其attrs
            print(title.string)
```

习近平主持召开重磅会议 重点研究这两件大事
李克强主持会议 确定政府工作报告重点任务分工
中央有关部门有诚心有行动听取香港各方面意见
北方沙尘天继续影响17省份
下半月扩散条件有利
喂药养羊、“瘦身”钢筋……这些黑幕被曝光！

上述过程中，会有少量的噪音数据，对比之后，我们可以通过一定的方法排除这些噪音数据

接下来，我们爬取要闻部分，即下图所示新闻


中新视频
探访武汉高龄患者康复驿站
美侨界驳“中国病毒”论
有爱公益视频 武警训练热血沸腾 男孩寄语医疗队
美国首家隔离酒店 学者谈欧洲防疫 武磊确诊感染
网络直播说相声 “隔离”火锅推出 校园消杀防疫
万亩桃林花开正盛 线上春茶开园节 探访老牌煤矿
学习强国 xuexi.cn
梦想从学习开始，事业从实践起步。
微视界
更多
严防境外疫情输入！北上广深进一步升级防控措施
警方通报武汉刑释人员进京事件：不追究黄某英法律责任
专家再谈血浆疗法：监测可能风险 亟需更多捐献
核安全局：海阳核电厂2号机组两起运行事件为0级事件
多支救援力量增援山西榆社山火火场 扑救仍在进行中
贵州江南等地有中到大雨 华北东北多地气温或创新高
湖北省委书记：武汉要有序扩展无疫社区居民活动空间
多地“花式接待”返乡援鄂医护人员 网友：这个可以有
关注前沿汽车资讯 就在中新网汽车频道
拼关键州！亿万富翁布隆伯格资助民主党1800万美元
外媒：东京奥运会组织方已开始草拟延期方案
中国专家医疗队受最高礼遇迎接 塞总统亲吻五星红旗
在美华人驳“中国病毒”论：别有用心 华人需积极应对
自私！疫情期间呆在家里
div.new_con_yw 380 × 1106
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
 == $0

<ul class="module_topcon_ul">
<li id="9133934">

严防境外疫情输入！北上广深进一步升级防控措施

<li id="9134010">…
<li id="9133894">…

In [59]:
```
# 要闻
important_news = soup.find(name="div", attrs={"class" :"new_con_yw"})

important_news = important_news.findAll(name="li")
for n in important_news:
    if n.string and len(n.attrs) == 1:  # 排除含有target的导航页面
        print(n.string)
#             <a href="/gj/2020/03-22/9133682.shtml">日本奥组委副主席：目前未到决定奥运延期的阶段<
#             <a href="http://auto.chinanews.com/" target="_blank">关注前沿汽车资讯 就在中新网汽车
```

```
英雄烈士公祭办法征意见：不得利用公祭从事犯罪活动
国务院任免国家工作人员：左力任司法部副部长
推广疫苗+筛查 距离“没有宫颈癌的未来”还有多远？
全国不到20人！如何缓解石窟寺考古人才断档之痛？
致盲性沙眼已被消灭 中国人群眼部疾病谱已发生重大变化
前2月主要经济指标呈大幅增长 2021中国经济开局怎么看
德法意也暂停接种阿斯利康疫苗 欧洲药监局将紧急开会
美意法德英发表联合声明：不会让叙冲突再延续十年
“乱港”两案同日开庭 黎智英一案进入结案阶段
闭门举行！东京奥组委宣布圣火出发仪式以无观众形式举行
涉以危险武器袭击警察 参与美国会暴乱的两名男子被捕
韩法院重启李在镕狱中审判 三星“失去的十年”或成现实
美边境人道危机：在押移民儿童数攀升 3000人超期羁押
A股小幅高开：沪指涨0.14%，医药板块活跃
透视美国强刺激：助力美国经济腾飞、薅全世界的羊毛？
新一线城市房租啥水平？杭州比肩一线城市、沈阳最低
花卉产业恢复迅速 鲜花消费每逢佳节必爆单如何破局？
美三大股指集体收涨 道指、标普500指数均创新高
“保险+期货”迎发展新机遇 为乡村产业撑起“保护伞”
余额吊胃口、投入是黑洞 预付费卡为何“清零”难？
广西回应“泡药沃柑”事件：对健康不构成影响
国产剧掀“耽改”热：“腐文化”出圈，青少年入坑
揭秘故宫太和殿：楠木从何而来？装修有何深意？
广州“巨无霸”违章公馆争议多年终被拆除 11人被追责
“冷板凳”“又难又穷”…如何摘掉基础学科旧“标签”
“两弹城”保护沉疴：院部面目全非 卖与企业后保护遇阻
小学生体测不合格不给毕业证 南通体测方案引关注
长大后我就成了你：98年“抱树女孩”当上人民警察
```

2.1 滚动新闻页面爬取

- 点击滚动，进入新的网址为https://www.chinanews.com/scroll-news/news1.html (https://www.chinanews.com/scroll-news/news1.html)，这里的新闻格式相对比较简单，我们可以先找到大的新闻框架（class= content_list的div），然后再定位到每一条新闻。
- 底部有页码栏，我们分别点击2和3，观察新打开的url，便可发现1-3页的url分别为：
  - https://www.chinanews.com/scroll-news/news1.html (https://www.chinanews.com/scroll-news/news1.html)
  - https://www.chinanews.com/scroll-news/news2.html (https://www.chinanews.com/scroll-news/news2.html)

- https://www.chinanews.com/scroll-news/news3.html (https://www.chinanews.com/scroll-news/news3.html)

便可发现每一页的url基本相同，只有数字变化的规律，所以我们通过数字，手动构造1-10的每一页的url，来进行每一页新闻的自动爬取

In [60]:
```
url_scroll = soup.find(name='ul',attrs={"class":"nav_navcon"}).find('a')['href'] #获取顶部导航栏扌
url_scroll = "https:" + url_scroll      # https://www.chinanews.com/scroll-news/news1.html

all_scroll_news = []

for i in range(1,11):
    url_scroll = url_scroll[:-6] + str(i) + ".html"        # 每一页构造url
    print("Page %d, get news form %s" % (i, url_scroll))


    html_scroll = requests.get(url_scroll)
    soup_scroll = BeautifulSoup(html_scroll.content,"html.parser")
    news = soup_scroll.find(name="div", attrs={"class" :"content_list"})
    news = news.findAll(name="div",attrs={"class":"dd_bt"})

    for n in news:
#         print(n.a.string)   # 爬取新闻标题
        all_scroll_news.append(n.a.string)
    print("len news:",len(all_scroll_news))
    print()
```

```
Page 1, get news form https://www.chinanews.com/scroll-news/news1.html (https://www.chinanews.com/scroll-news/news1.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 125

Page 2, get news form https://www.chinanews.com/scroll-news/news2.html (https://www.chinanews.com/scroll-news/news2.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 250

Page 3, get news form https://www.chinanews.com/scroll-news/news3.html (https://www.chinanews.com/scroll-news/news3.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 375

Page 4, get news form https://www.chinanews.com/scroll-news/news4.html (https://www.chinanews.com/scroll-news/news4.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 500

Page 5, get news form https://www.chinanews.com/scroll-news/news5.html (https://www.chinanews.com/scroll-news/news5.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 625

Page 6, get news form https://www.chinanews.com/scroll-news/news6.html (https://www.chinanews.com/scroll-news/news6.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 750

Page 7, get news form https://www.chinanews.com/scroll-news/news7.html (https://www.chinanews.com/scroll-news/news7.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.
```

```
len news: 875

Page 8, get news form https://www.chinanews.com/scroll-news/news8.html (https://www.chinanews.com/scroll-news/news8.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 1000

Page 9, get news form https://www.chinanews.com/scroll-news/news9.html (https://www.chinanews.com/scroll-news/news9.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 1125

Page 10, get news form https://www.chinanews.com/scroll-news/news10.html (https://www.chinanews.com/scroll-news/news10.html)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

len news: 1250
```

因为涉及到编码问题，上述单元格运行会产生 Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER. 的警告

但对我们的结果没有影响。如果不想产生类似警告，可以修改下面两行加注释的代码

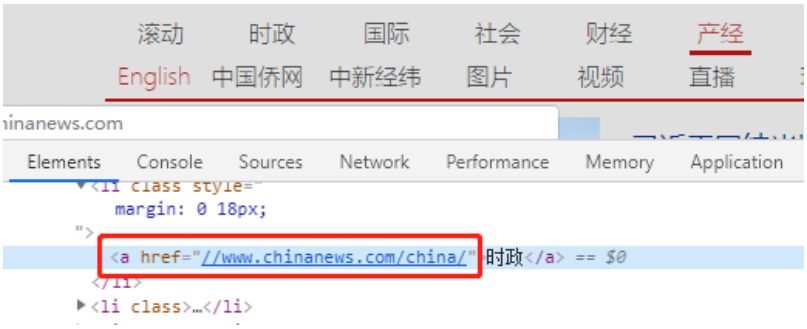
In [61]:
```
# html_scroll = requests.get(url_scroll)
# soup_scroll = BeautifulSoup(html_scroll.content,"html.parser",from_encoding="iso-8859-1") # 编码
# news = soup_scroll.find(name="div", attrs={"class" :"content_list"})
# news = news.findAll(name="div",attrs={"class":"dd_bt"})

# for n in news:
#     print(n.a.string.encode("iso-8859-1").decode("utf-8"))     # 编码转换
```

In [62]:
```
len(all_scroll_news),all_scroll_news[:10] # 查看数据
```

Out[62]:
```
(1250,
 ['东京奥组委确认火炬接力将如期举行 不对普通观众开放',
  '驻尼日利亚使馆为接种国产疫苗人员赴华提供便利',
  '阿根廷一男子抢劫华人超市 被店员和顾客合力擒住',
  '王晶、陈百祥等发声 十几位香港演艺人士挺“爱国者治港”',
  '无线网络信号不好还总掉线？我国专家有了新解决方案',
  '安徽庐剧“接棒”传承焕生机 专业院团青年演员占比近半',
  '探索全新合作模式 松下携手中国石化建立冬奥合作示范店',
  '意大利暂停阿斯利康疫苗 疫情期间抗焦虑药物销量大增',
  '云南畹町边检站破获一起特大毒品案 缴毒近12公斤',
  '3月下半月中国大部扩散条件较为有利 空气质量以优良为主'])
```

2.2 获取各板块头条新闻以及热点新闻

对于这类每页有大致相同结构的网页，虽然每个板块url不能直接构造，但是我们可以借助其属性href获取每一页的url



In [63]:
```
all_news = []
navbar = soup.find(name='ul',attrs={"class":"nav_navcon"}).findAll("a")
for nav in navbar[1:14]: # 取第一栏
    print(nav.string)
    if nav['href'][:5]!="http:":
        url_nav = "http:" + nav['href']  # 获取改板块的url
    print(url_nav)
    html_nav = requests.get(url_nav)
    soup_nav = BeautifulSoup(html_nav.content,"html.parser")

    count = 0  # 控制打印新闻条数
    for n in soup_nav.findAll("li") + soup_nav.findAll("em") + soup_nav.findAll("h1"):  # 寻找新
        if n.a and n.a.string and len(n.a.string) > 7:
            all_news.append(n.a.string)
            if count < 3:
                print(n.a.string)
                count += 1
    print()
print(len(all_news))
```

```
时政
http://www.chinanews.com/china/ (http://www.chinanews.com/china/)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

陕西省首个黄河流域生态环境司法保护基地揭牌
跨越十七载：一群台湾人与四川凉山的“达祖”情缘
中国城市人均公园绿地面积达14.8平方米

国际
http://www.chinanews.com/world/ (http://www.chinanews.com/world/)

Some characters could not be decoded, and were replaced with REPLACEMENT CHARACTER.

• 意大利暂停阿斯利康疫苗 疫情期间抗焦虑药物销量大增
• 美圣地亚哥汽车冲撞行人事故致3死6伤 司机已被捕
• 土耳其向叙利亚投放军事设备修筑防御工事 俄表示担忧

社会
http://www.chinanews.com/society/ (http://www.chinanews.com/society/)
```

In [64]:
```
# 我们生成一个新的数据文本，并将所有新闻标题写入新文件
with open('data/all_news.txt', 'w') as f:
    for i in range(len(all_news)):
        f.write(str(all_news[i])+'\n')
f.close()
```

实例3：根据API获取股票金融数据

采集数据

下面的例子中采集股票数据，需要安装pandas-datareader

```
conda install pandas-datareader
```

加载模块如下

```
from pandas_datareader import data
```

In [65]:
```
#  使用Yahoo Finance的API获取四个公司的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

codes = ['AAPL', 'IBM', 'MSFT', 'GOOG']  # 四个股票
all_stock = {}
for ticker in codes:
    all_stock[ticker] = data.get_data_yahoo(ticker,start='1/1/2018')  # 默认从2010年1月起始

volume = pd.DataFrame({tic: data['Volume'] for tic, data in all_stock.items()})
open = pd.DataFrame({tic: data['Open'] for tic, data in all_stock.items()})
high = pd.DataFrame({tic: data['High'] for tic, data in all_stock.items()})
low =  pd.DataFrame({tic: data['Low'] for tic, data in all_stock.items()})
close =  pd.DataFrame({tic: data['Close'] for tic, data in all_stock.items()})
price = pd.DataFrame({tic: data['Adj Close'] for tic, data in all_stock.items()}) # 已调整或者复权
```

In [66]:
```
all_stock['AAPL'].head() # 显示前5行数据
```

Out[66]:

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-02	43.075001	42.314999	42.540001	43.064999	102223600.0	41.380238
2018-01-03	43.637501	42.990002	43.132500	43.057499	118071600.0	41.373032
2018-01-04	43.367500	43.020000	43.134998	43.257500	89738400.0	41.565216
2018-01-05	43.842499	43.262501	43.360001	43.750000	94640000.0	42.038452
2018-01-08	43.902500	43.482498	43.587502	43.587502	82271200.0	41.882305

In [67]:
```
all_stock['AAPL'].tail() # 显示后5行数据
```

Out[67]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-03-09	122.059998	118.790001	119.029999	121.089996	129159600.0	121.089996
2021-03-10	122.169998	119.449997	121.690002	119.980003	111760400.0	119.980003
2021-03-11	123.209999	121.260002	122.540001	121.959999	102753600.0	121.959999
2021-03-12	121.169998	119.160004	120.400002	121.029999	87963400.0	121.029999
2021-03-15	124.000000	120.430000	121.410004	123.989998	92590555.0	123.989998

In [68]:
```
price.head()
```

Out[68]:

Date	AAPL	IBM	MSFT	GOOG
2018-01-02	41.380238	132.039154	82.194328	1065.000000
2018-01-03	41.373032	135.668625	82.576843	1082.479980
2018-01-04	41.565216	138.416412	83.303658	1086.400024
2018-01-05	42.038452	139.092651	84.336464	1102.229980
2018-01-08	41.882305	139.931534	84.422516	1106.939941

In [69]:
```
all_stock['AAPL'].head()
```

Out[69]:

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-02	43.075001	42.314999	42.540001	43.064999	102223600.0	41.380238
2018-01-03	43.637501	42.990002	43.132500	43.057499	118071600.0	41.373032
2018-01-04	43.367500	43.020000	43.134998	43.257500	89738400.0	41.565216
2018-01-05	43.842499	43.262501	43.360001	43.750000	94640000.0	42.038452
2018-01-08	43.902500	43.482498	43.587502	43.587502	82271200.0	41.882305

In [70]:
```
AAPL = all_stock['AAPL']
len(AAPL)
AAPL.head()
```

Out[70]:

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-02	43.075001	42.314999	42.540001	43.064999	102223600.0	41.380238
2018-01-03	43.637501	42.990002	43.132500	43.057499	118071600.0	41.373032
2018-01-04	43.367500	43.020000	43.134998	43.257500	89738400.0	41.565216
2018-01-05	43.842499	43.262501	43.360001	43.750000	94640000.0	42.038452
2018-01-08	43.902500	43.482498	43.587502	43.587502	82271200.0	41.882305

In [71]:
```
# 为了以后更容易导入数据，我们生成一个新的csv数据文本，并将所有数据行写入新文件
AAPL.to_csv('data/AAPL-0.csv')
```

In [72]:
```
# 有时网络访问不稳定，因此读入已经保存的AAPL股票数据
import numpy as np
import pandas as pd
f = 'data/AAPL-0.csv'

data = pd.read_csv(f, index_col='Date')  # 使用date列作为行索引
data.index = pd.to_datetime(data.index)  # 将字符串索引转换成时间索引
AAPL = data
print(len(AAPL))
AAPL.tail() # 显示后5行数据
```

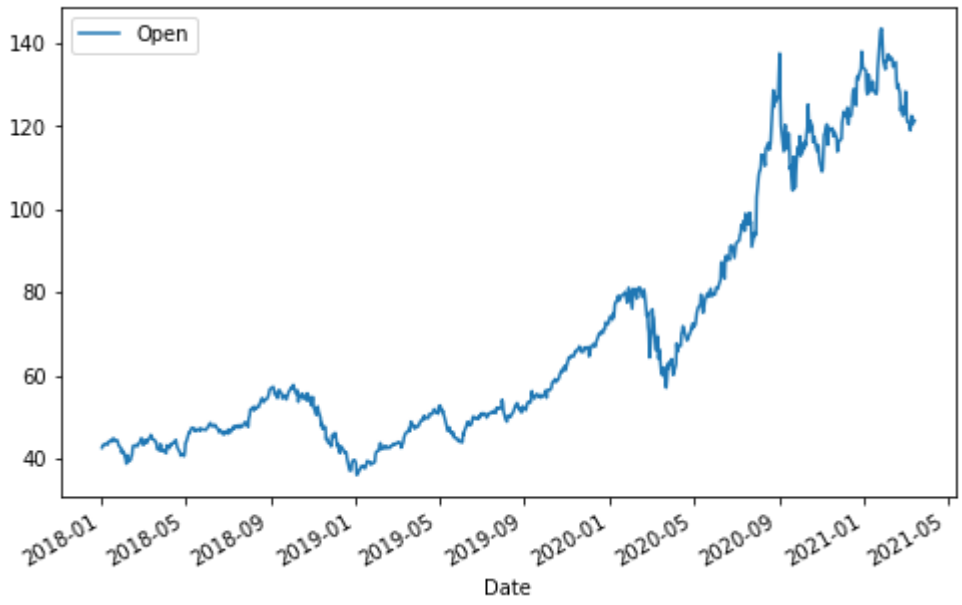
805

Out[72]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-03-09	122.059998	118.790001	119.029999	121.089996	129159600.0	121.089996
2021-03-10	122.169998	119.449997	121.690002	119.980003	111760400.0	119.980003
2021-03-11	123.209999	121.260002	122.540001	121.959999	102753600.0	121.959999
2021-03-12	121.169998	119.160004	120.400002	121.029999	87963400.0	121.029999
2021-03-15	124.000000	120.430000	121.410004	123.989998	92590555.0	123.989998

In [73]:
```
# 包含两种趋势的典型股价图表绘图
AAPL[['Open']].plot(figsize=(8,5))
```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x7f950afbc690>



使用Yahoo Finance的API获取沪深股市的股票数据

In [74]:
```
#  使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
#from pandas_datareader import data
import pandas_datareader as pdr

# 获取
Maotai = pdr.get_data_yahoo('600519.SS') # 茅台股票代码+沪市
Maotai.head()
```

Out[74]:

Date	High	Low	Open	Close	Volume	Adj Close
2016-03-17	234.100006	228.300003	230.000000	229.559998	2882095.0	212.597107
2016-03-18	232.259995	225.779999	232.000000	226.000000	7056894.0	209.300140
2016-03-21	232.020004	226.649994	226.660004	231.979996	3854745.0	214.838272
2016-03-22	238.179993	231.500000	232.259995	234.490005	4176661.0	217.162796
2016-03-23	237.000000	232.679993	235.199997	234.970001	2141206.0	217.607315

In [75]:
```
Maotai.tail()
```

Out[75]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-03-09	2000.000000	1900.180054	1955.000000	1936.989990	8226581.0	1936.989990
2021-03-10	1999.869995	1967.000000	1977.000000	1970.010010	5117174.0	1970.010010
2021-03-11	2079.989990	1961.479980	1976.989990	2048.000000	5676897.0	2048.000000
2021-03-12	2077.000000	2002.010010	2070.000000	2026.000000	4032251.0	2026.000000
2021-03-16	1997.989990	1965.000000	1974.930054	1983.660034	1411958.0	1983.660034

In [76]:
```
#  使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data
import pandas_datareader as pdr

# 获取
PUFA = data.get_data_yahoo('600000.SS') # 浦发银行股票代码600000+沪市SS
PUFA.head()
```

Out[76]:

Date	High	Low	Open	Close	Volume	Adj Close
2016-03-17	12.720279	12.202797	12.454545	12.377622	98149211.0	10.799910
2016-03-18	12.587412	12.342657	12.440559	12.475524	80062794.0	10.885332
2016-03-21	12.762237	12.433566	12.433566	12.629370	60062190.0	11.019567
2016-03-22	13.027972	12.601398	12.650349	12.692307	62577959.0	11.074484
2016-03-23	12.832167	12.629370	12.699300	12.678321	39798036.0	11.062279

In [77]:
```
PUFA.tail()
```

Out[77]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-03-09	10.90	10.65	10.78	10.73	83584061.0	10.73
2021-03-10	10.79	10.63	10.76	10.72	50512161.0	10.72
2021-03-11	10.97	10.76	10.77	10.90	76816962.0	10.90
2021-03-12	11.03	10.82	10.89	10.90	79235629.0	10.90
2021-03-16	11.24	11.05	11.05	11.09	30280655.0	11.09

In [78]:
```
#  使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

# 获取探路者股票代码是：300005，深市SZ
EXP = data.get_data_yahoo('300005.SZ') # 探路者股票代码是：300005，创业板股票代码以300打头
EXP.head()
```

Out[78]:

Date	High	Low	Open	Close	Volume	Adj Close
2016-03-17	11.173333	10.593333	10.660000	11.140000	21816462.0	10.808415
2016-03-18	11.886666	11.153333	11.166666	11.733333	30138343.0	11.384086
2016-03-21	12.126666	11.666666	11.940000	12.100000	30252804.0	11.739841
2016-03-22	12.320000	11.833333	11.906666	12.086666	25129914.0	11.726903
2016-03-23	12.400000	11.866666	12.166666	12.146666	21871434.0	11.785117

In [79]:
```
EXP.tail()
```

Out[79]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-03-09	7.06	6.67	7.00	6.85	12553447.0	6.85
2021-03-10	6.88	6.70	6.85	6.85	9440310.0	6.85
2021-03-11	6.90	6.68	6.89	6.72	11100550.0	6.72
2021-03-12	6.85	6.60	6.71	6.82	9638400.0	6.82
2021-03-16	6.75	6.60	6.72	6.63	2910200.0	6.63