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. 网络新闻标题采集，解析网页内容
3. 获取Web数据（股票数据），并进行股票数据的分析和处理，保存磁盘

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
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
data.cov() # 数据的协方差矩阵
data.corr() # 数据的Pearson相关系数矩阵
```

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.columns
```

Out[6]:

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='objec
t')
```

In [7]:

```
data.columns.names  # 此时每个列没有name
```

Out[7]:

```
FrozenList([None])
```

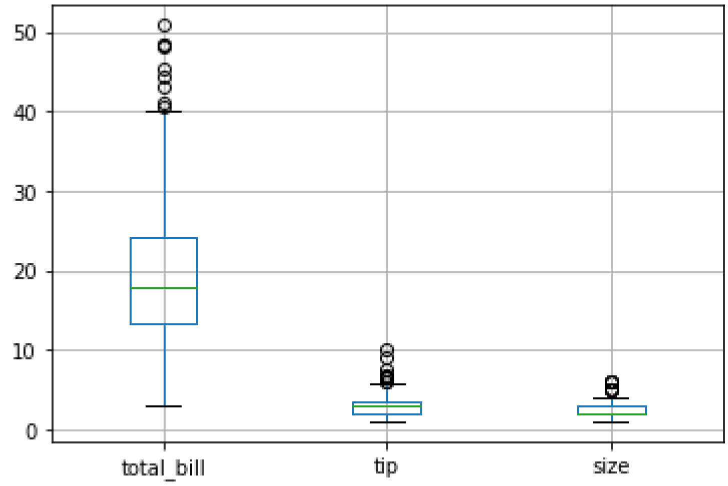
In [8]:

```
# 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[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247be79e788>
```

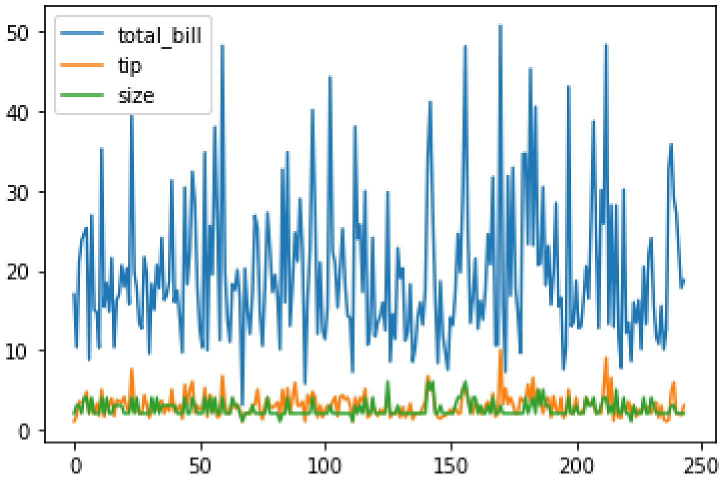


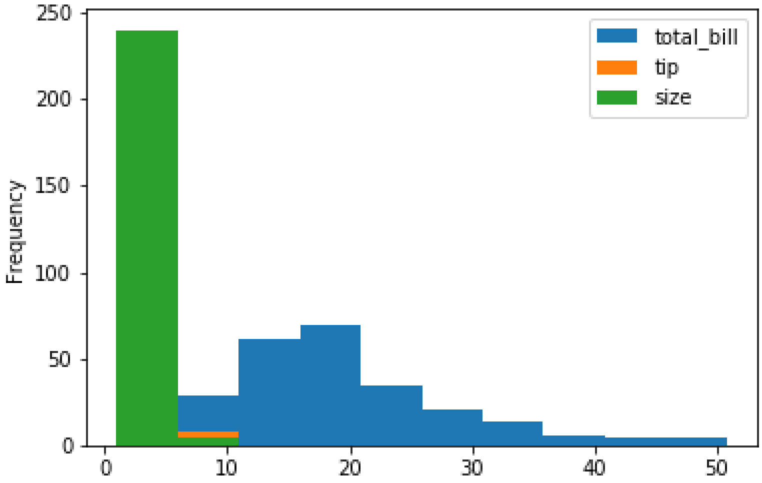
In [9]:

```
data.plot.line()   # 线图
data.plot.hist()   # 直方图
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247be92a308>
```



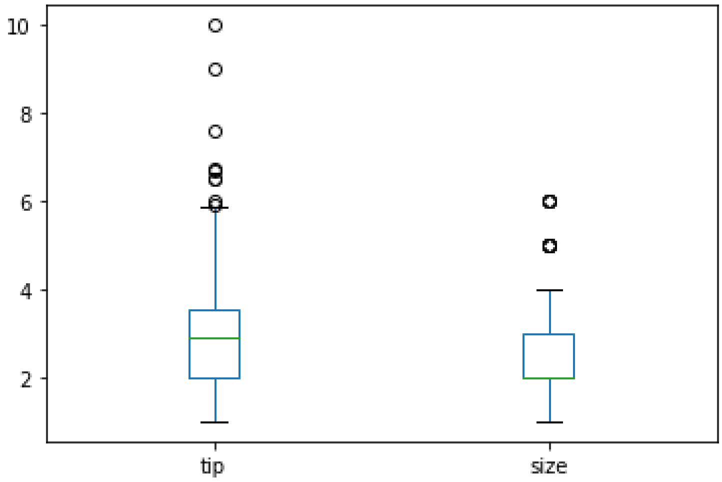


In [10]:

```
# 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[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247bea00748>
```



In [11]:

```
# 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 [12]:

```
# 相关性分析
data.corr()  # method : {’pearson’, ’kendall’, ’spearman’}, 默认pearson
```

Out[12]:

	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 [13]:

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

Out[13]:

	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 [14]:

```
data.head()
```

Out[14]:

	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 [15]:

```
# 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[15]:

		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 [16]:

```
data2.tail()
```

Out[16]:

		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 [17]:

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

Out[17]:

	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 [18]:

```
# 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[18]:

	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 [19]:

```
# 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 [20]:

```
# 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 [21]:

```
# 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[21]:

		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 [22]:

```
# 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 [23]:

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

Out[23]:

	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 [24]:

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

Out[24]:

		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 [25]:

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

Out[25]:

		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 [26]:

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

Out[26]:

		total_bill	tip	size	tip_pct
day	**time**				
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 [27]:

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

Out[27]:

```
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 [28]:

```
# 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[28]:

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

In [29]:

```
# 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[29]:

		total_bill	tip	size	tip_pct
sex	**time**				
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 [30]:

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

Out[30]:

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

In [31]:

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

Out[31]:

```
sex
Female    0.166491
Male      0.157651
Name: tip_pct, dtype: float64
```

In [32]:

```
# 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[32]:

```
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 [33]:

```
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[33]:

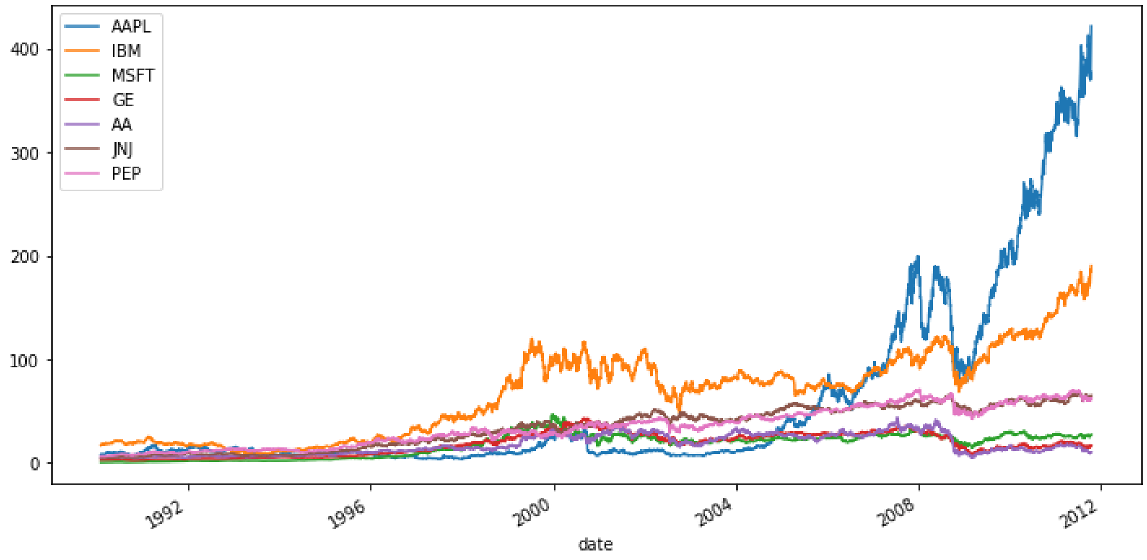
	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
date									
2011-10-10	10.09	388.81	16.14	186.62	64.43	26.94	61.87	1194.89	76.28
2011-10-11	10.30	400.29	16.14	185.00	63.96	27.00	60.95	1195.54	76.27
2011-10-12	10.05	402.19	16.40	186.12	64.33	26.96	62.70	1207.25	77.16
2011-10-13	10.10	408.43	16.22	186.82	64.23	27.18	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

In [34]:

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

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247c0746cc8>
```



In [35]:

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

Out[35]:

```
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 [36]:

```
# 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 [37]:

```
# 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[37]:

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	X
AA	1.000000	0.101313	0.916804	0.600211	0.685752	0.776796	0.634679	0.846861	0.567
AAPL	0.101313	1.000000	0.142381	0.749037	0.651564	0.423274	0.741942	0.410813	0.78
GE	0.916804	0.142381	1.000000	0.681659	0.717824	0.875398	0.652904	0.935598	0.58
IBM	0.600211	0.749037	0.681659	1.000000	0.902894	0.871615	0.885029	0.835484	0.85
JNJ	0.685752	0.651564	0.717824	0.902894	1.000000	0.846906	0.970478	0.845401	0.92
MSFT	0.776796	0.423274	0.875398	0.871615	0.846906	1.000000	0.781791	0.949715	0.73
PEP	0.634679	0.741942	0.652904	0.885029	0.970478	0.781791	1.000000	0.816477	0.96
SPX	0.846861	0.410813	0.935598	0.835484	0.845401	0.949715	0.816477	1.000000	0.76
XOM	0.567025	0.781369	0.581171	0.855195	0.925600	0.730557	0.964278	0.761077	1.00

In [38]:

```
data.head()
```

Out[38]:

	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 [39]:

```
data.tail()
```

Out[39]:

	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 [40]:

```
# 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[40]:

	AAPL
date	
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 [41]:

```
# 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[41]:

	AAPL	Return
date		
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 [42]:

```
# 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[42]:

	AAPL	Return	Return2
date			
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 [43]:

```
# 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[43]:

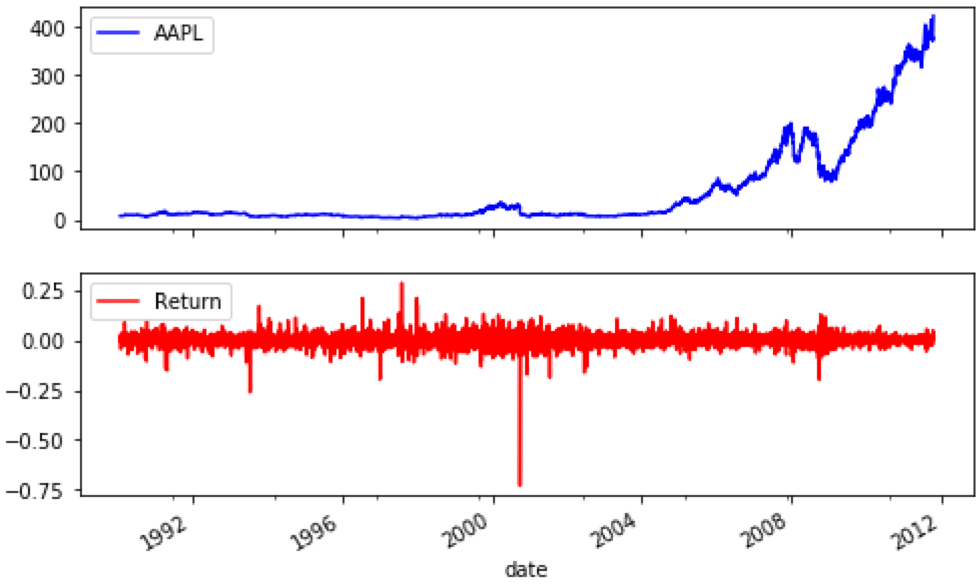
	AAPL	Return
date		
2011-10-10	388.81	0.050128
2011-10-11	400.29	0.029098
2011-10-12	402.19	0.004735
2011-10-13	408.43	0.015396
2011-10-14	422.00	0.032685

In [44]:

```
# 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[44]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000247C0857388>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000247C0881C88>],
      dtype=object)
```



In [45]:

```
# 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[45]:

	AAPL	42d	252d
date			
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 [46]:

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

Out[46]:

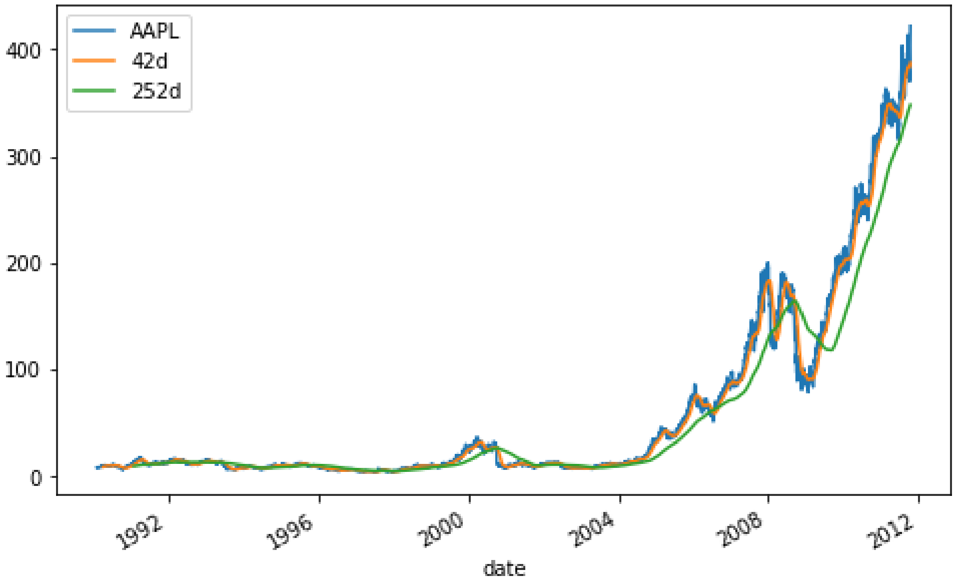
	AAPL	42d	252d
date			
1990-02-01	7.86	NaN	NaN
1990-02-02	8.00	NaN	NaN
1990-02-05	8.18	NaN	NaN
1990-02-06	8.12	NaN	NaN
1990-02-07	7.77	NaN	NaN

In [47]:

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

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247c09e5308>
```



In [48]:

```
# 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[48]:

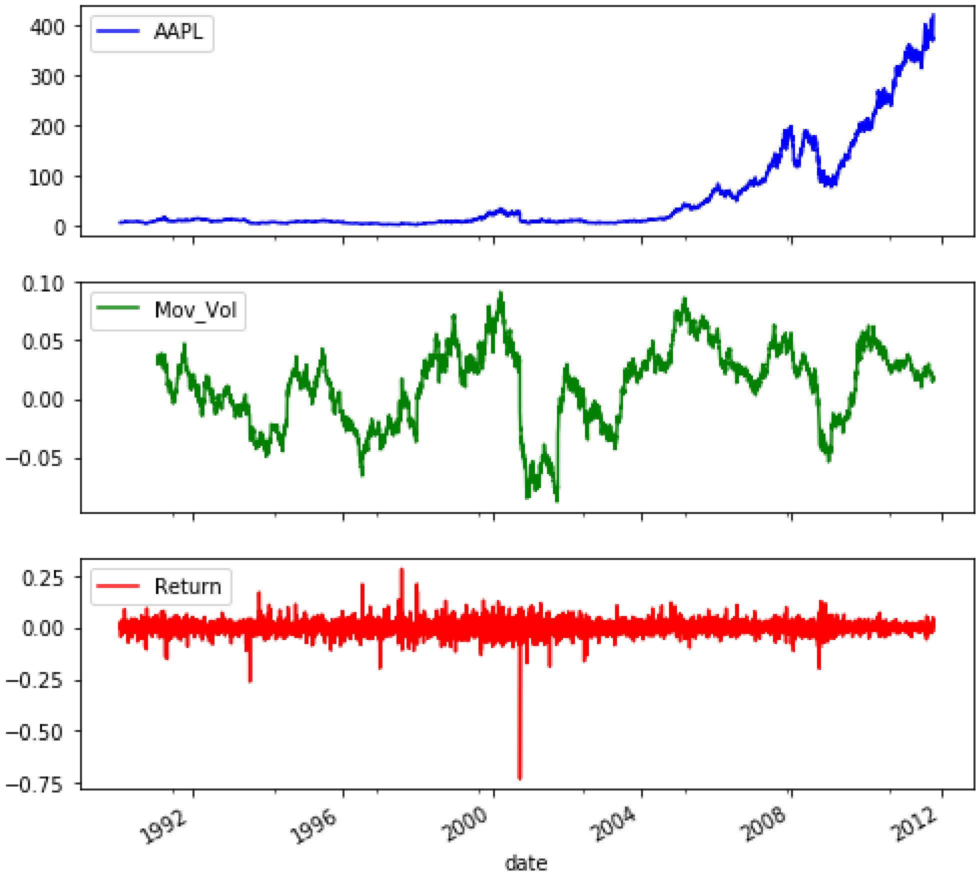
```
date
2011-10-10    0.017317
2011-10-11    0.018475
2011-10-12    0.018437
2011-10-13    0.018953
2011-10-14    0.018474
Name: Mov_Vol, dtype: float64
```

In [49]:

```
# 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[49]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000247C09CD7C8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000247C0A64588>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000247C0AA5808>],
      dtype=object)
```

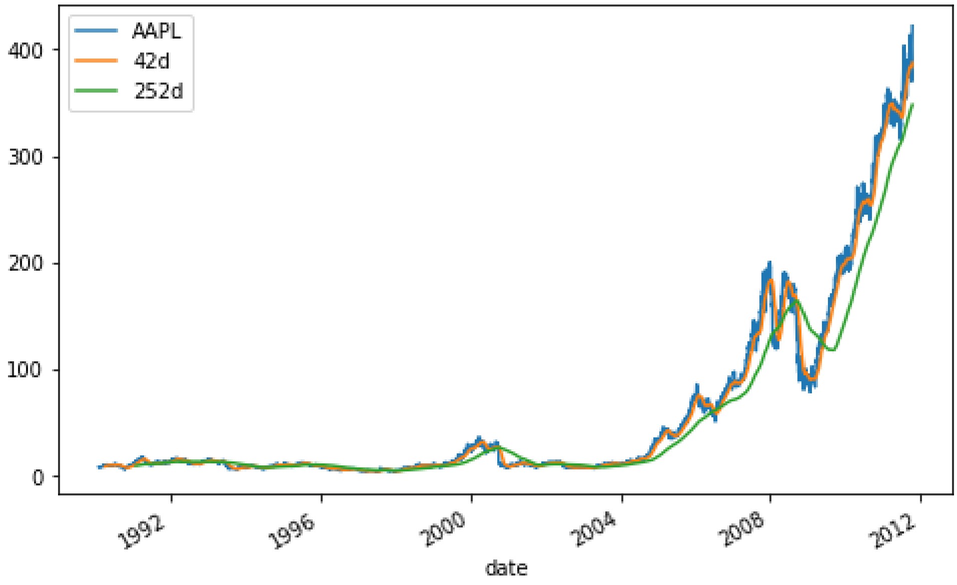


In [50]:

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

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247c0d989c8>
```



In [51]:

```
df.tail()
```

Out[51]:

	AAPL	Return	42d	252d	Mov_Vol
date					
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 [52]:

```
# 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 [ ]:

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

In [54]:

```
# 加载访问url资源的库
from bs4 import BeautifulSoup
import requests
```

In [55]:

```
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 [56]:

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

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当行为</a>  # 目标数据
#         <a class="ptv" href="//www.chinanews.com/shipin/spfts/20200321/2659.shtml">回看</a>  # 噪音数据
        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==2)
            print(title.string)
```

```
全球战疫时刻 习近平连发慰问电传递中国态度
31省份新增39例境外输入
应勇辞去上海市市长职务 龚正任上海市代市长
多国“逼宫”百年奥运或首次推迟举办
何去何从
```

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

In [57]:

```
# 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>       # 目标数据
#             <a href="http://auto.chinanews.com/" target="_blank">关注前沿汽车资讯 就在中新网汽车频道</a>    # 噪声数据
```

```
美方称中方策划掩盖疫情？外交部：贼喊捉贼把戏拙劣
图解：学生和家长们！这份开学防疫指南请收好默记
当前主要防输入性风险 张文宏：正常生活可逐步恢复
倒查时间跨度达20年之久 内蒙古涉煤反腐风暴前夜
第二批中国援意医疗专家组组长：传染源是最大问题
林郑月娥：25日起所有非香港居民乘飞机抵港不准入境
华侨悉尼拍vlog：街头口罩罕见 居民抢购生活用品
台湾“国标舞女王”刘真未等到换心病逝 享年44岁
以总理提议组建“紧急联合政府” 被质疑“真诚度”
结婚纪念日妻子被隔离 美九旬丈夫举标语：爱你67年
美第三轮紧急经济计划参议院遇阻 规模超1.8万亿美元
NASA科学家验证“月球导航”可能性 能接收GPS信号
广西柳州螺蛳粉产业园复工复产， 一起在线催货
去年末中国金融业机构总资产318.69万亿 同比增8.6%
两部门：克服疫情影响 千方百计防范化解农产品卖难
北京：3家用人单位拖欠农民工工资被列入“黑名单”
未成年人网游消费纠纷突出 实名认证形同虚设
你守护生命，我守护你！医疗队背后强大的后勤团队
修建墓穴燃放炮竹引发山西榆社山火 嫌犯已被刑拘
五大连池通报“防疫卡口发生冲突”：值守人员受处分
官方回应杭州未成年留学生未集中隔离：已赴隔离点
内蒙古鄂尔多斯被刺医生伤情平稳 无生命危险
妻子帮因公殉职的他完成遗愿：遗体器官救治8名患者
“无症状感染者”为何没纳入确诊病例？专家这样解释
```

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 [58]:

```
url_scroll = soup.find(name='ul',attrs={"class":"nav_navcon"}).find('a')['href'] #获取顶部导航栏排第一的滚动页面的url
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

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

len news: 125

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

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

len news: 250

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

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

len news: 375

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

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

len news: 500

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

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

len news: 625

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

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

len news: 750

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


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

len news: 875

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

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

len news: 1000

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

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

len news: 1125

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

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

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 [59]:

```
# html_scroll = requests.get(url_scroll)
# soup_scroll = BeautifulSoup(html_scroll.content,"html.parser",from_encoding="iso-8859-1") # 编码转换
# 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 [60]:

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

Out[60]:

```
(1250,
 ['广州隔离酒店增至76个 日新增入境隔离上千人',
  '日本政府相关人士：若国际奥委会做出东京奥运会延期决定，日方将予以同意',
  '福建漳州公安机关速破一起故意伤害致死案',
  '加澳两国呼吁推迟奥运会 日首相表态或考虑推迟',
  '中国抗“疫”观察：“善待湖北人”的错误和正确“打开方式”',
  '西安：进港国际航班逐人检测 严把境外人员入京关口',
  '海关总署：对来自重点防控国家的交通工具100%实施登临检疫',
  '国家移民管理局：外国人不如实填报信息等不准入境',
  '新冠疫情下 穷人处境凸显美国社会的残酷',
  '世界气象日：破解这些气象传言'])
```

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 [ ]:

```
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")

    for n in soup_nav.findAll("li") + soup_nav.findAll("em") + soup_nav.findAll("h1"):  # 寻找
新闻特定
        if n.a and n.a.string and len(n.a.string) > 7:
            print(n.a.string)
            all_news.append(n.a.string)
    print()
print(len(all_news))
```

In [62]:

```
# 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 [63]:

```
#  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()}) # 已调整或者复权后的收盘价,能比较真实反映股票的表现
```

```
C:\Anaconda3\lib\site-packages\pandas_datareader\compat\__init__.py:7: FutureWarni
ng: pandas.util.testing is deprecated. Use the functions in the public API at pand
as.testing instead.
  from pandas.util.testing import assert_frame_equal
```

In [64]:

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

Out[64]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2018-01-02	172.300003	169.259995	170.160004	172.259995	25555900.0	166.804016
2018-01-03	174.550003	171.960007	172.529999	172.229996	29517900.0	166.774963
2018-01-04	173.470001	172.080002	172.539993	173.029999	22434600.0	167.549622
2018-01-05	175.369995	173.050003	173.440002	175.000000	23660000.0	169.457214
2018-01-08	175.610001	173.929993	174.350006	174.350006	20567800.0	168.827820

In [65]:

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

Out[65]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-03-16	259.079987	240.000000	241.949997	242.210007	80605900.0	242.210007
2020-03-17	257.609985	238.399994	247.509995	252.860001	81014000.0	252.860001
2020-03-18	250.000000	237.119995	239.770004	246.669998	75058400.0	246.669998
2020-03-19	252.839996	242.610001	247.389999	244.779999	67964300.0	244.779999
2020-03-20	251.830002	228.000000	247.179993	229.240005	100257000.0	229.240005

In [66]:

```
price.head()
```

Out[66]:

	AAPL	IBM	MSFT	GOOG
Date				
2018-01-02	166.804016	139.365463	83.029594	1065.000000
2018-01-03	166.774963	143.196335	83.415993	1082.479980
2018-01-04	167.549622	146.096573	84.150192	1086.400024
2018-01-05	169.457214	146.810333	85.193497	1102.229980
2018-01-08	168.827820	147.695770	85.280426	1106.939941

In [67]:

```
all_stock['AAPL'].head()
```

Out[67]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2018-01-02	172.300003	169.259995	170.160004	172.259995	25555900.0	166.804016
2018-01-03	174.550003	171.960007	172.529999	172.229996	29517900.0	166.774963
2018-01-04	173.470001	172.080002	172.539993	173.029999	22434600.0	167.549622
2018-01-05	175.369995	173.050003	173.440002	175.000000	23660000.0	169.457214
2018-01-08	175.610001	173.929993	174.350006	174.350006	20567800.0	168.827820

In [68]:

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

Out[68]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2018-01-02	172.300003	169.259995	170.160004	172.259995	25555900.0	166.804016
2018-01-03	174.550003	171.960007	172.529999	172.229996	29517900.0	166.774963
2018-01-04	173.470001	172.080002	172.539993	173.029999	22434600.0	167.549622
2018-01-05	175.369995	173.050003	173.440002	175.000000	23660000.0	169.457214
2018-01-08	175.610001	173.929993	174.350006	174.350006	20567800.0	168.827820

In [69]:

```
# 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 [70]:

```
# 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
```

```
558
```

Out[70]:

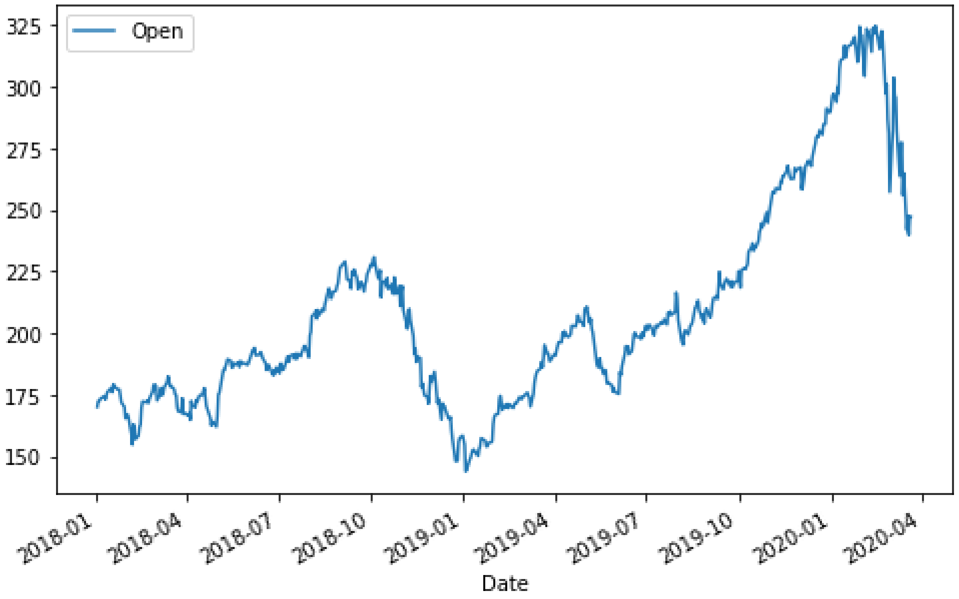
	High	Low	Open	Close	Volume	Adj Close
Date						
2020-03-16	259.079987	240.000000	241.949997	242.210007	80605900.0	242.210007
2020-03-17	257.609985	238.399994	247.509995	252.860001	81014000.0	252.860001
2020-03-18	250.000000	237.119995	239.770004	246.669998	75058400.0	246.669998
2020-03-19	252.839996	242.610001	247.389999	244.779999	67964300.0	244.779999
2020-03-20	251.830002	228.000000	247.179993	229.240005	100257000.0	229.240005

In [71]:

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

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247c4477dc8>
```



使用Yahoo Finance的API获取沪深股市的股票数据

In [72]:

```
#  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[72]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2015-03-25	177.363998	174.600006	175.455002	175.255005	4646433.0	161.346069
2015-03-26	181.727005	174.544998	175.955002	180.509003	7381577.0	166.183090
2015-03-27	180.509003	175.408997	180.000000	176.863998	4415904.0	162.827377
2015-03-30	180.863998	175.544998	176.981995	178.882004	6979652.0	164.685211
2015-03-31	182.518005	176.382004	178.800003	178.145004	9694406.0	164.006714

In [73]:

```
Maotai.tail()
```

Out[73]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-03-17	1078.000000	1011.119995	1055.00000	1045.099976	7935912.0	1045.099976
2020-03-18	1060.000000	1007.989990	1040.00000	1007.989990	7345988.0	1007.989990
2020-03-19	1015.000000	960.099976	993.98999	996.000000	10226507.0	996.000000
2020-03-20	1043.000000	1011.000000	1011.00000	1035.280029	6013919.0	1035.280029
2020-03-23	1035.280029	991.520020	1000.00000	1019.000000	4847210.0	1019.000000

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

Out[74]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2015-03-25	10.9930	10.5944	10.9091	10.6573	474374374.0	9.101697
2015-03-26	10.8951	10.5175	10.5944	10.7413	403747219.0	9.173435
2015-03-27	10.8741	10.6084	10.7133	10.7413	311990218.0	9.173435
2015-03-30	11.3287	10.7972	10.8741	11.2028	697043003.0	9.567574
2015-03-31	11.5944	11.0070	11.5245	11.0420	706490433.0	9.430243

In [75]:

```
PUFA.tail()
```

Out[75]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-03-17	10.69	10.29	10.68	10.43	42139008.0	10.43
2020-03-18	10.50	10.20	10.40	10.21	33176646.0	10.21
2020-03-19	10.34	9.86	10.19	9.94	55707113.0	9.94
2020-03-20	10.18	9.96	10.12	10.09	42609358.0	10.09
2020-03-23	9.95	9.82	9.95	9.84	28286030.0	9.84

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

# 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[76]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2015-03-25	21.360001	20.333300	21.273300	21.053301	31169518.0	20.425993
2015-03-26	20.766701	19.040001	20.666700	19.426701	32285590.0	18.847860
2015-03-27	20.166700	19.033300	19.033300	19.833300	26836396.0	19.242344
2015-03-30	19.946699	18.866699	19.733299	19.320000	27484341.0	18.744339
2015-03-31	19.993299	19.073299	19.333300	19.639999	24465148.0	19.054802

In [77]:

```
EXP.tail()
```

Out[77]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-03-17	4.00	3.82	3.96	3.91	8885138.0	3.91
2020-03-18	3.97	3.79	3.95	3.81	10559252.0	3.81
2020-03-19	3.87	3.70	3.79	3.85	9974147.0	3.85
2020-03-20	3.88	3.78	3.86	3.85	8526071.0	3.85
2020-03-23	3.77	3.58	3.77	3.61	10287300.0	3.61

作业 1：

1）新闻标题数据下载及分析

- 下载2周的新闻数据（尽可能多）
- 进行学习过的数据预处理技术（重复记录，噪音数据清理等）
- 分析并观察各种板块新闻数据的分布等

注意：留存采集的新闻数据，后续作业需要使用自己采集的数据

2）新闻标题数据下载及分析

- 从Yahoo！Finance下载美交所各种题材股票（阿里，百度，京东等）
- 从Yahoo！Finance下载沪深交所各种题材股票（上证股票是股票代码后面加上.ss，深证股票是股票代码后面加上.sz）
- 分析并观察各种题材股票(例如保险类，新能源类，互联网相关)的各种统计情况、趋势、相关性分析等
- 上证股票是股票代码后面加上.ss，深证股票是股票代码后面加上.sz; 香港为 0001.hk;加拿大股指代码：cnu.to;新西兰股指代码为.nz;新加坡股指代码为.si;台湾股指代码为.tw

注意：留存采集的股票数据，后续作业需要使用自己采集的数据