《数据挖掘导论》实验 3: 可视化分析实验

一、实验目的：

（1）了解matplotlib的绘图组件
（2）掌握pandas常用的绘图方法

二、实验环境：

（1）Anaconda3 开发环境
（2）IDE是Jupyter notebook
（3）使用的库有numpy，pandas，matplotlib

三、实验内容：

1. 使用 pandas 读入数据。

1） 使用 pandas 的 read_csv 函数

```
read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_col=None, usecols=None)
```

- 参数

  - filepath_or_buffer：数据文件的路径
  - sep：自定义分隔符，默认为逗号','，
  - header：用作列名的行号。默认为 0（第一行），如果没有 header 行就应该设置为 None
  - names：用于结果的列名列表，结合 header=None
  - index_col：用作行索引的列标号或列名。可以使单个名称/数字组成的列表（层次化索引）。
  - usecols：一维素组，素组元素可以是整数表示字段索引号，也可以字符串表示字段名，返回素组指定的列。
  - parse_dates：True 表示将字符串解析成时间对象

2. 认识 matplotlib 的绘图对象

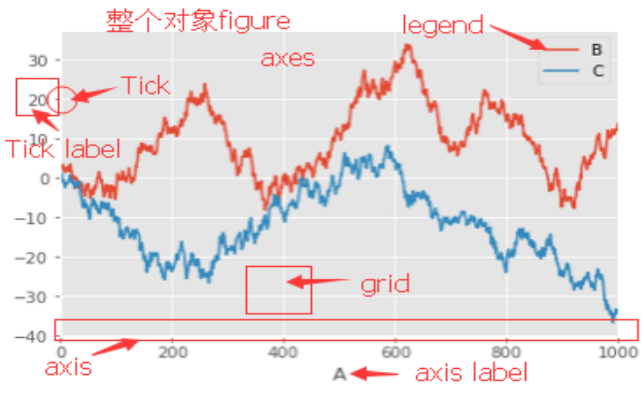
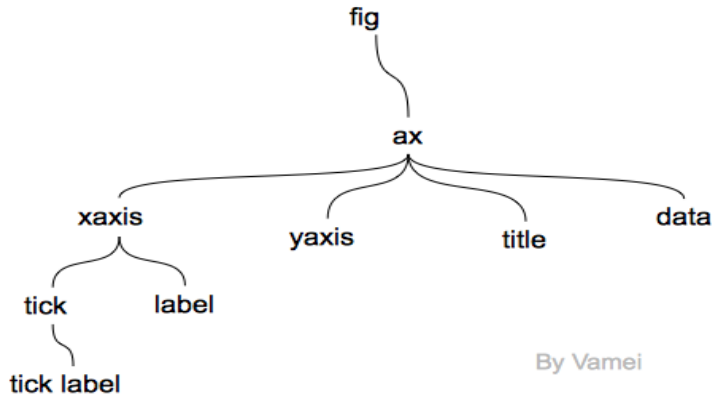


figure 是整个绘图对象，axes 可理解为一个画板，可以在上面画各种图形，legend 可以显示各个曲线的名称，axis 可理解为是一个 x 轴或者 y 轴的对象，Tick 是 axis 上的刻度，axis 有名称，每一个刻度也有名称。grid 参数控制是否绘制网格线。他们的包含关系如下图所示：



图片来源：http://www.cnblogs.com/vamei/archive/2013/01/30/2879700.html

3. Pandas 绘图函数

Pandas 绘图函数是对 matplotlib 绘图函数的一个包装，使得绘图更方便。

1) plot 函数

plot(x=None, y=None, kind='line', figsize=None, subplots=False, grid=None, legend=True, title=None)

- 参数
  - x：x 轴的坐标，默认为 None，指的是用 dataframe 的 index 作为坐标
  - y：y 轴的坐标，默认为 None，指的每个字段认为是一个 y 的坐标，如绘制所有曲线。
  - kind：字符串，str
    - 'line' : 默认是绘制折线图
    - 'bar' : 绘制柱状图
    - 'barh' : 横着的柱状图
    - 'hist' : 直方图
    - 'box' : 箱图
    - 'kde' : 密度图(通过计算可能会产生观测数据的连续概率分布的估计而产生的)
    - 'density' : same as 'kde'
    - 'area' : 面积图
    - 'pie' : 饼图
    - 'scatter' : 散点图
    - 'hexbin' : 绘制高密度散点图
  - figsize：指绘图尺寸，默认为 None
  - subplots：是否绘制多个子图，默认为 False
  - grid：是否绘制网格线，默认为 False
  - legend：是否显示曲线的名称，默认为 True

- title：绘图的名称，默认为 None
- 返回的是 axis 对象

2) scatter_matrix 函数

pandas 库中独立的一个方法，用于绘制散布图。

scatter_matrix(frame, alpha=0.5, figsize=None, ax=None, grid=False, diagonal='hist', marker='.')

□ 参数

- frame：是一个 DataFrame 对象
- alpha 设置透明度，0 到 1 的浮点数
- figsize：图片尺寸
- ax：可以设置绘图的对象，默认为 None，则会新建一个 axes 对象。
- grid：判断是否设置网格线
- diagonal：设置对角线上图的类型，默认是'hist'，表示绘制直方图。
- marker：设置散点的类型，默认是‘.’，表示绘制实心点。
- 返回的是 axis 对象

3) andrews_curves 函数

pandas 绘制调和曲线的函数在 pandas.tools.plotting 中。
andrews_curves(frame, class_column)

□ 参数

- frame，DataFrame 对象
- class_column，字符串表示字段名，指示类标所在的列。
- 返回的是 axis 对象

4. 案例一，股票数据勘探

1) 读取股票数据

有四只美国股票价格从 2003 年到 2011 年的数据，数据的格式如下，第一行是字段名字，第一列是日期：

	AAPL	MSFT	XOM	SPX
2003-01-02	7.40	21.11	29.22	909.03

使用如下函数读入数据，返回的是一个 DataFrame 对象，parse_dates 为 true 会把时间解析成 datetime 时间对象，index_col 是指将第一列（索引为 0 的一列），即时间那一列作为索引，索引类型为 DatetimeIndex。

```
close_px_all = pd.read_csv('stock_px.csv', parse_dates=True, index_col=0)
```

使用 head()函数查看前 5 行数据：

```
close_px_all.head()
```

结果如下表所示：

	AAPL	MSFT	XOM	SPX
2003-01-02	7.40	21.11	29.22	909.03
2003-01-03	7.45	21.14	29.24	908.59
2003-01-06	7.45	21.52	29.96	929.01
2003-01-07	7.43	21.93	28.95	929.93
2003-01-08	7.28	21.31	28.83	909.93

2) 绘制折线图

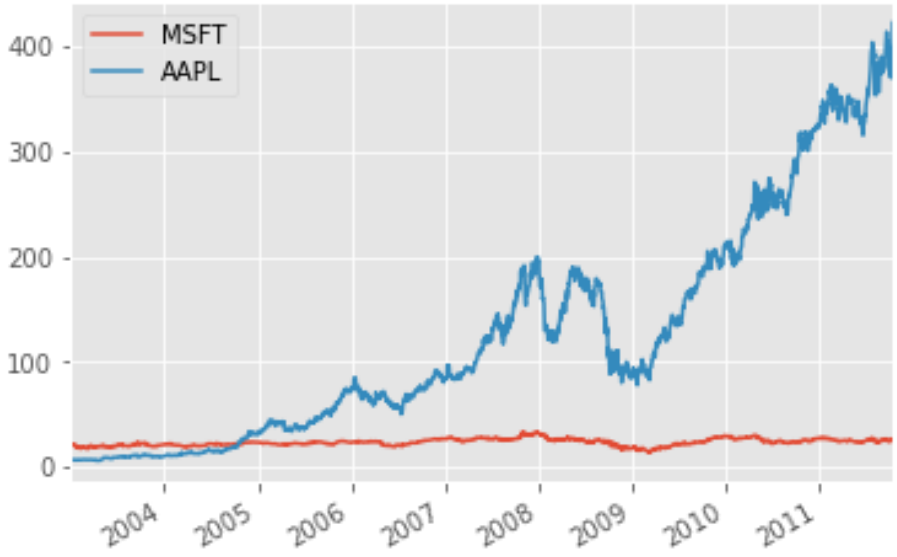
绘制股票价格变化折线图
导入绘图的库

```
import matplotlib.pyplot as plt
import matplotlib
#使绘图内嵌到 notebook 中
%matplotlib inline
```

使用 DataFrame 对象的 plot 函数绘制'MSFT'和'AAPL'股票的价格曲线

```
close_px_all.plot(y=['MSFT', 'AAPL'], kind='line')
```

结果如下图所示：



3) 绘制柱状图

绘制股票每年平均价格柱状图
从 DatetimeIndex 中可以取出每个时间的年份

```
# 在读取表格的时候，已经将时间的数据转换为
# DatetimeIndex 对象了,并作为 DataFrame 的 index，现在可以
# 取出每个时间的 year 属性值，用 print 语句查看 DataFrame
```

```
# 的 index 的每个时间对象的年份属性。

print(close_px_all.index.year)

# 输出为：
# Int64Index([2003, ... 2011],  dtype='int64', length=2214)
```

使用 dataFrame 对象的 groupby 方法统计每只股票每年平均股票价格

```
# 根据 close_px_all.index.year 返回的年份作来进行分组，用
# mean 函数求每组的股票的平均值
meanPrice = close_px_all.groupby(close_px_all.index.year).mean()
```

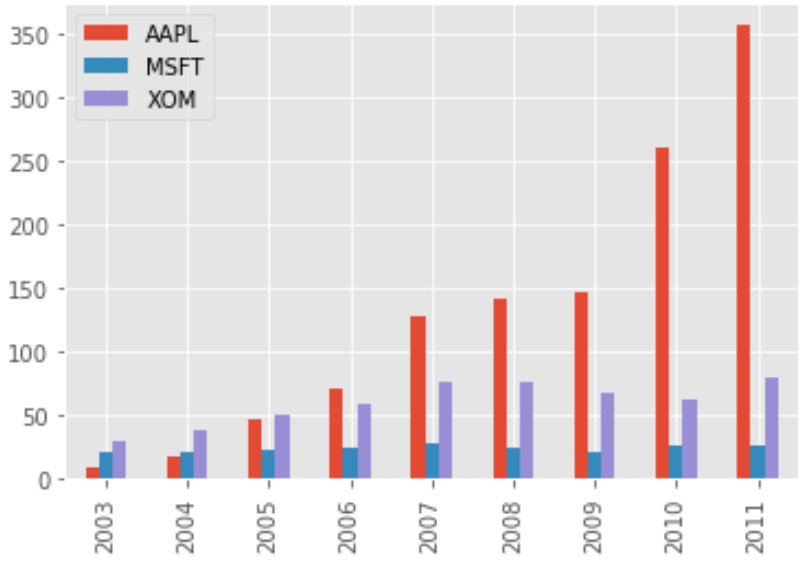
统计的平均价格如下表所示：

	AAPL	MSFT	XOM	SPX
2003	9.272619	20.59512	30.21111	965.2275
2004	17.76389	21.85044	38.87544	1130.649
2005	46.67595	23.07242	51.04548	1207.229
2006	70.81064	23.75936	58.45841	1310.462
2007	128.2739	27.90442	75.76713	1477.184
2008	141.979	24.76059	76.52597	1220.042
2009	146.8141	21.8854	67.12496	948.0464
2010	259.8425	26.26262	63.06798	1139.966
2011	356.5268	25.82593	79.04266	1276.093

绘制'AAPL', 'MSFT'和'XOM'股票每年平均价格的柱状图

```
meanPrice.plot(kind='bar', y=['AAPL', 'MSFT', 'XOM'])
```

结果如下图所示：

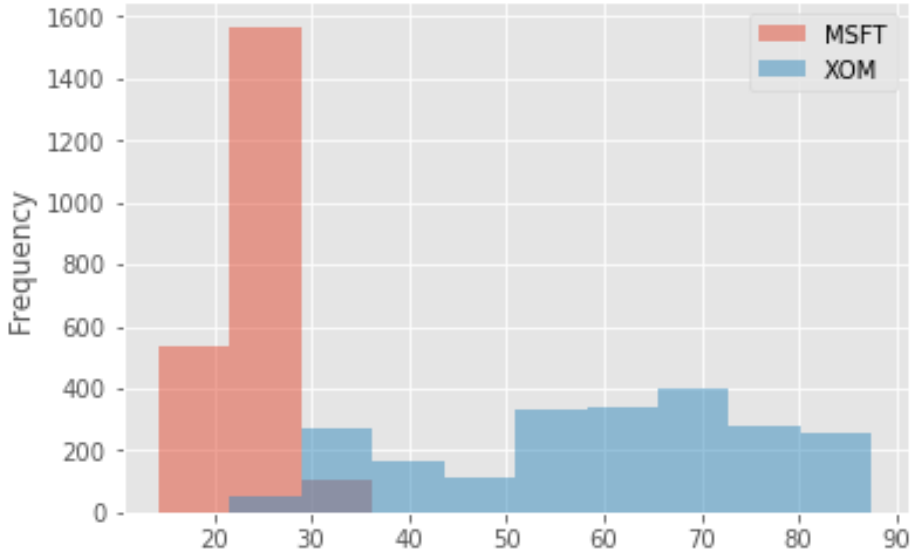


4）绘制直方图

绘制'MSFT'和'XOM'股票价格的直方图，可以了解到股票价格在各个价格区间的次数。

```
#  这里的 alpha 参数是用于设置透明度的
close_px_all.plot(y=['MSFT', 'XOM'], kind='hist', alpha=0.5)
```

结果如下图所示：

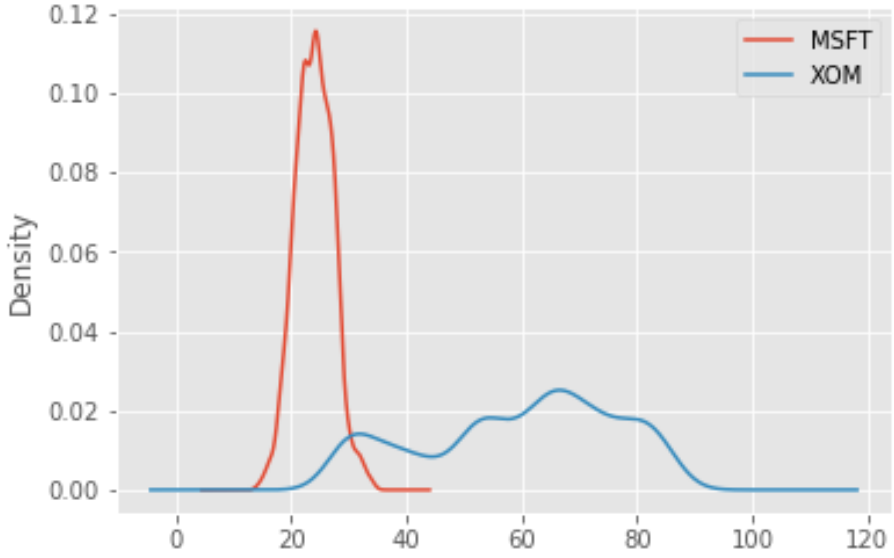


5）绘制密度图

例绘制'MSFT'和'XOM'股票价格的密度图，即价格的连续概率分布的估计，可以看出查看股票价格在各个价格区间的频率分布。

使用 DataFrame 的 plot 函数， 设置 kind 的值为'kde'。

close_px_all.plot(y=['MSFT', 'XOM'], kind='kde')



6）绘制饼图

例绘制四只股票在 2010 年，根据每个月平均值占 12 个月价格总和的比例绘制饼图。

筛选出 2010 年的数据

```
year2010 = close_px_all.iloc[close_px_all.index.year==2010]
```

求每个月平均值

```
year2010_month_mean =year2010.groupby(year2010.index.month).mean()
```

每个月平均值占 12 个月价格总和的比例

```
year2010_rate = year2010_month_mean / year2010_month_mean.sum()
```

结果如下表所示：

	AAPL	MSFT	XOM	SPX
1	0.066837	0.091866	0.086061	0.082183
2	0.063967	0.0866	0.083143	0.079665
3	0.071887	0.089573	0.084752	0.084265
4	0.080812	0.093446	0.087069	0.087576
5	0.080913	0.086242	0.080659	0.082291
6	0.084029	0.078175	0.077782	0.079241
7	0.082035	0.077105	0.075542	0.078981
8	0.080868	0.076178	0.077807	0.079528
9	0.088117	0.076035	0.079026	0.082073
10	0.096832	0.077855	0.083937	0.085694
11	0.10026	0.081227	0.090334	0.087691
12	0.103445	0.085698	0.093887	0.09081

使用 DataFrame 的 plot 函数， 设置 kind 的值为’pie’。

新建 4 个 axes 画图对象

```
fig, ax = plt.subplots(2,2,figsize=(10,10))
```

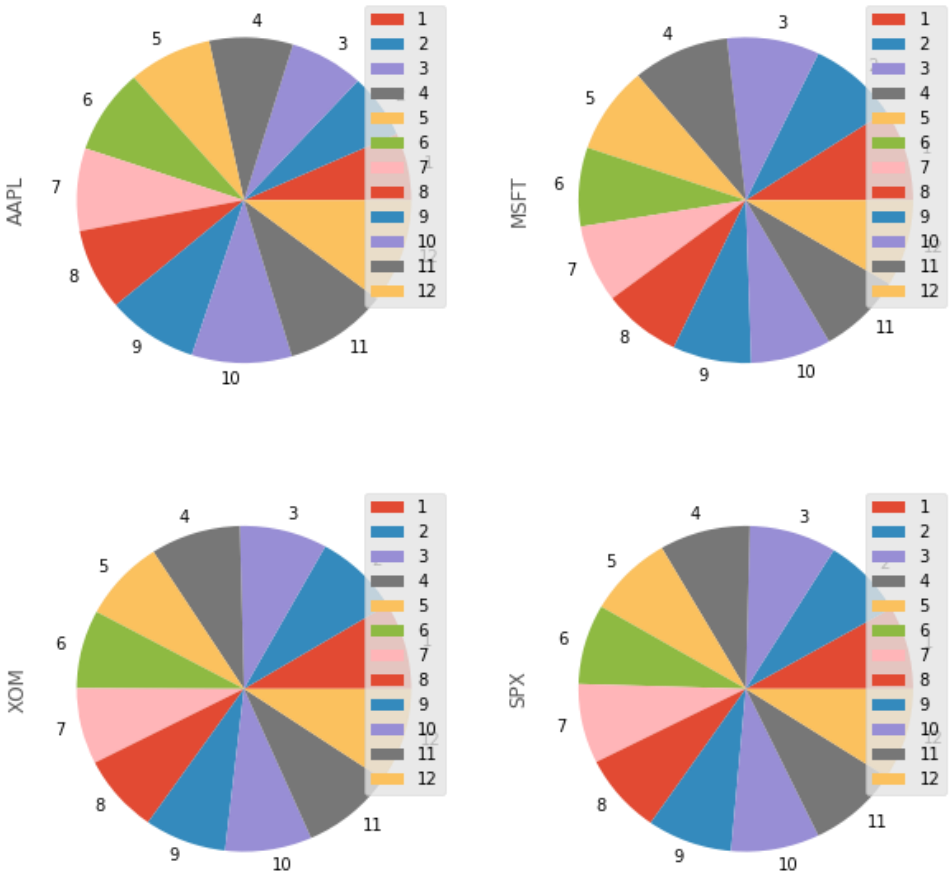
将画图数组改成列表形式

```
ax_list = [ax[0,0], ax[0,1], ax[1,0], ax[1,1]]
```

绘制饼图，参数 ax 设置绘图的 axes 对象，subplots 为 True 设置绘制多个子图。

```
year2010_rate.plot( kind='pie', ax=ax_list, subplots=True)
```

结果如下图所示：



5. 案例二，鸢尾花数据勘探

1) 读取鸢尾花数据

鸢尾花数据的格式如下表所示，第一行是字段名字，花萼的长，花萼的宽，花瓣的长，花瓣的宽和花的类别，其中花的类别有'setosa', 'versicolor', 'virginica'。

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
5.1	3.5	1.4	0.2	setosa

使用如下函数读入数据，返回的是一个 DataFrame 对象。

```
iris = pd.read_csv('iris.csv')
```

使用 head()函数查看前 5 行数据：

```
iris.head()
```

前 5 行数据如下表所示

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa

2) 绘制散点图

根据花瓣的长和宽绘制散点图

导入绘图的库

```
import matplotlib.pyplot as plt
import matplotlib
#使绘图内嵌到 notebook 中
%matplotlib inline
```

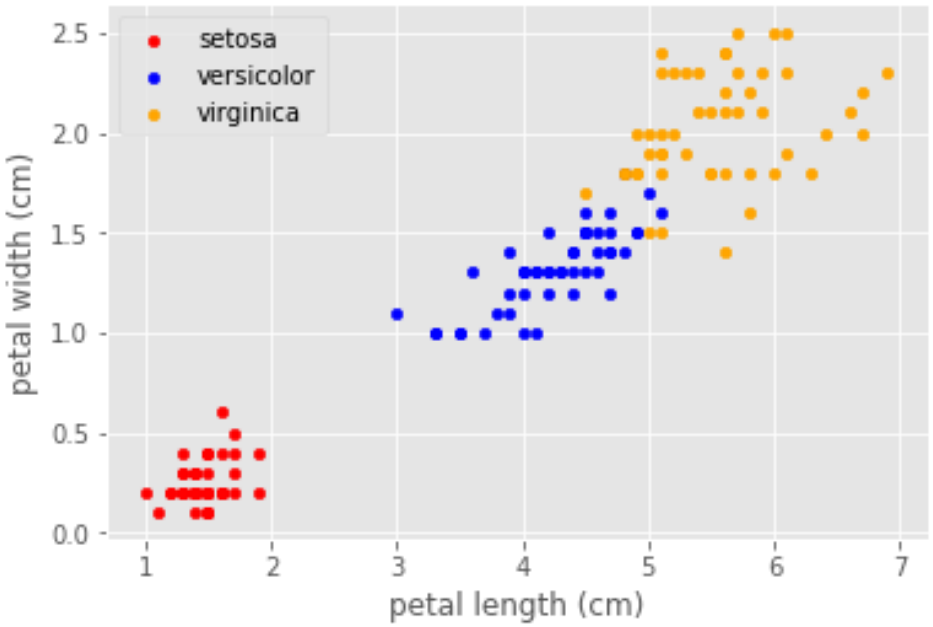
使用 DataFrame 对象的 plot 函数根据花瓣的长和宽绘制散点图，参数 x 是设置用作 x 轴坐标的字段，参数 y 设置用作 y 轴坐标的字段，参数 kind 设置绘图的类型，参数 c 设置颜色，ax 设置画板（默认为 None），label 设置散点的名称。

```
ax = iris[iris['class']=='setosa'].plot(x='petal length (cm)', y='petal
width (cm)', kind='scatter', c='red', ax=None, label='setosa')

iris[iris['class']=='versicolor'].plot(x='petal length (cm)', y='petal width
(cm)', kind='scatter', c='blue' , ax=ax, label='versicolor')

iris[iris['class']=='virginica'].plot(x='petal length (cm)', y='petal width
(cm)', kind='scatter', c='orange', ax=ax, label='virginica')
```

结果如下图所示，从下图来看，使用花瓣数据对花的区分度挺好的：



3) 绘制散布图

绘制散布图需要用到 pandas 的 scatter_matrix 函数。

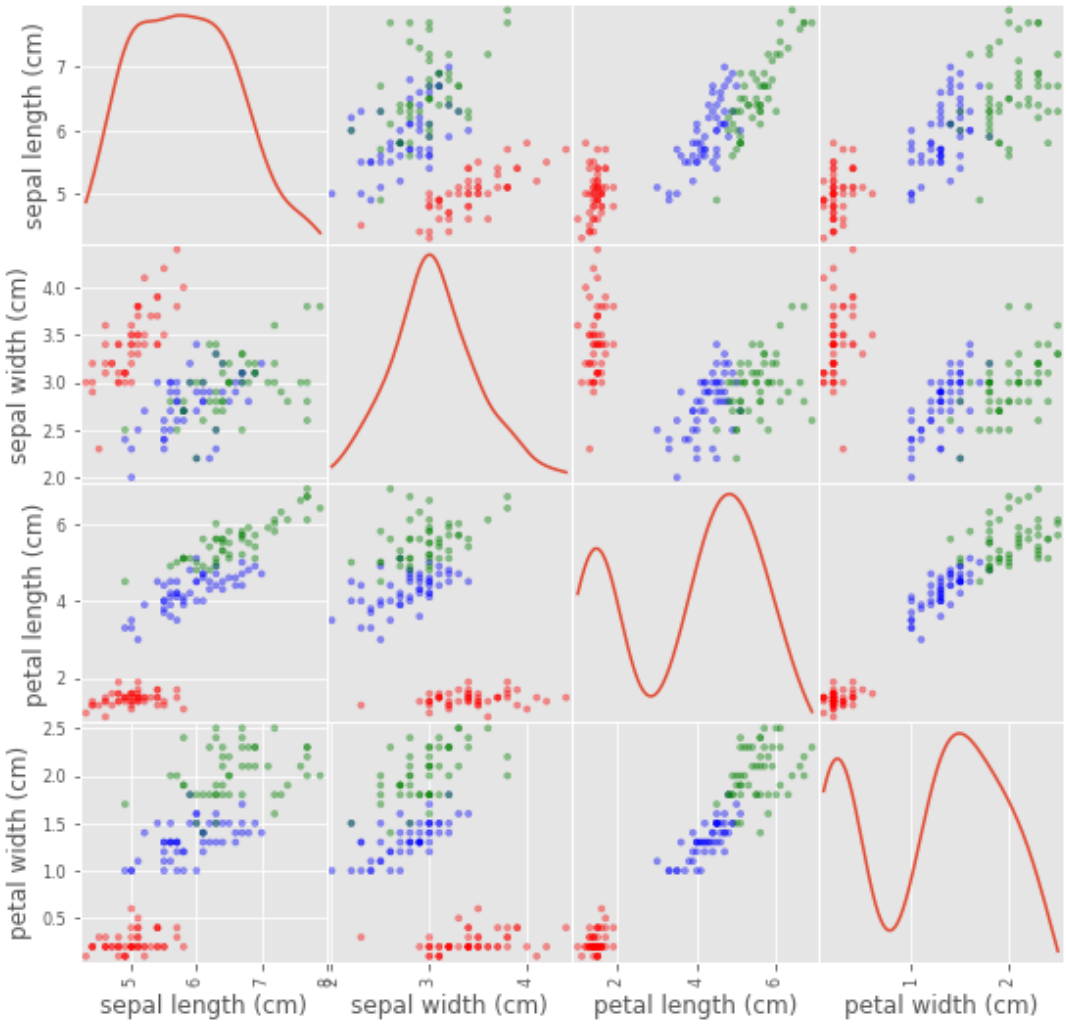
设置一个颜色字典，键是花的种类，值是颜色

```
color = {'setosa':'red', 'versicolor':'blue', 'virginica':'green'}
```

iris.iloc[:,:-1]是指取出除了最后一列外的数据，figsize 参数设置绘图大小，diagonal 设置对角线上绘制的图的类型，lambda x: color[x]是一个匿名函数，根据参数返回字典中对应的值，iris['class'].apply(lambda x: color[x])返回各个类别对应的颜色。

```
pd.scatter_matrix(iris.iloc[:,:-1], figsize=(9, 9), diagonal='kde', marker='o',
s=40, alpha=0.4, c=iris['class'].apply(lambda x: color[x]))
```

使用该函数绘制的散布图如下所示：



4) 绘制调和曲线图

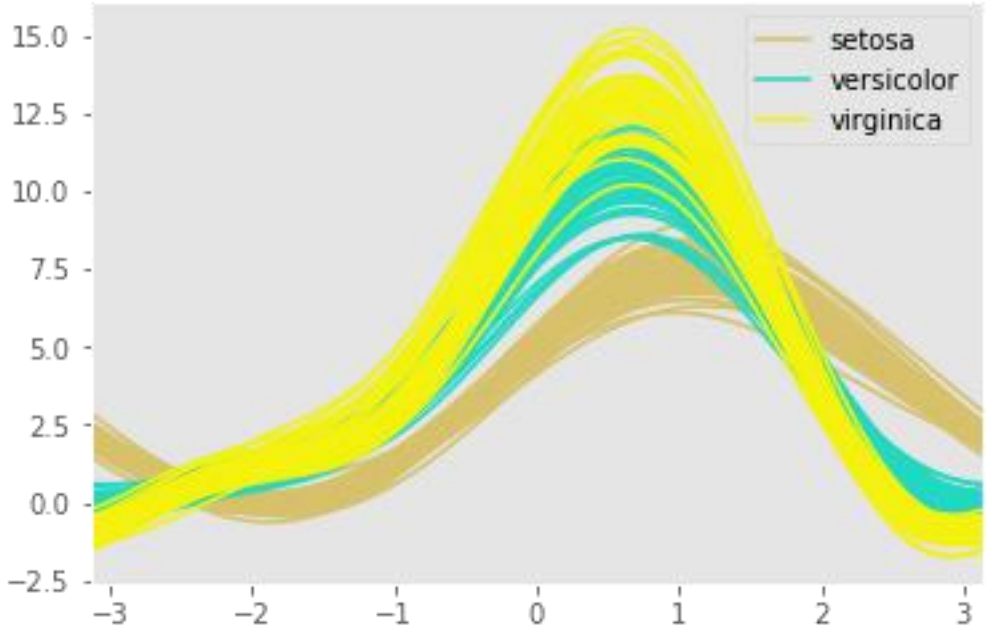
导入函数

```
from pandas.plotting import andrews_curves
```

使用 andrews_curves 绘制调和曲线，第一个参数是 DataFrame 对象，第二个参数是 DataFrame 对象中存放类别的列的名字。

```
andrews_curves(iris, 'class')
```

结果如下图所示：


15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−3
−2
−1
0
1
2
3
setosa
versicolor
virginica