数据分析 -- 数据可视化

数据可视化

- **数据分析**是一个探索性的过程，通常从特定的问题开始。它需要好奇心、寻找答案的欲望和很好的韧性，因为这些答案并不总是容易得到的。
- **数据可视化**，即数据的可视化展示。有效的可视化可显著减少受众处理信息和获取有价值见解所需的时间。
- 数据分析和数据可视化这两个术语密不可分。在实际处理数据时，数据分析先于可视化输出，而可视化分析又是呈现有效分析结果的一种好方法。

数据可视化

■**数据可视化**：是关于数据视觉表现形式的科学技术研究。其中，这种数据的视觉表现形式被定义为“一种以某种概要形式抽提出来的信息，包括相应信息单位的各种属性和变量”。

■数据可视化主要是借助于图形化手段，清晰有效地传达与沟通信息。

数据可视化 --相关标准库和扩展库

■Matplotlib

- Matplotlib是Python 的绘图库，依赖于numpy模块（和其他一些模块），可以绘制多种形式的图形，包括线图、直方图、饼状图、散点图、误差线图等等，图形质量可满足出版要求，是数据可视化的重要工具。

- 注意：pip install matplotlib 时有时由于下载超时而报错，多试几次

绘制带有中文标签和图例的图

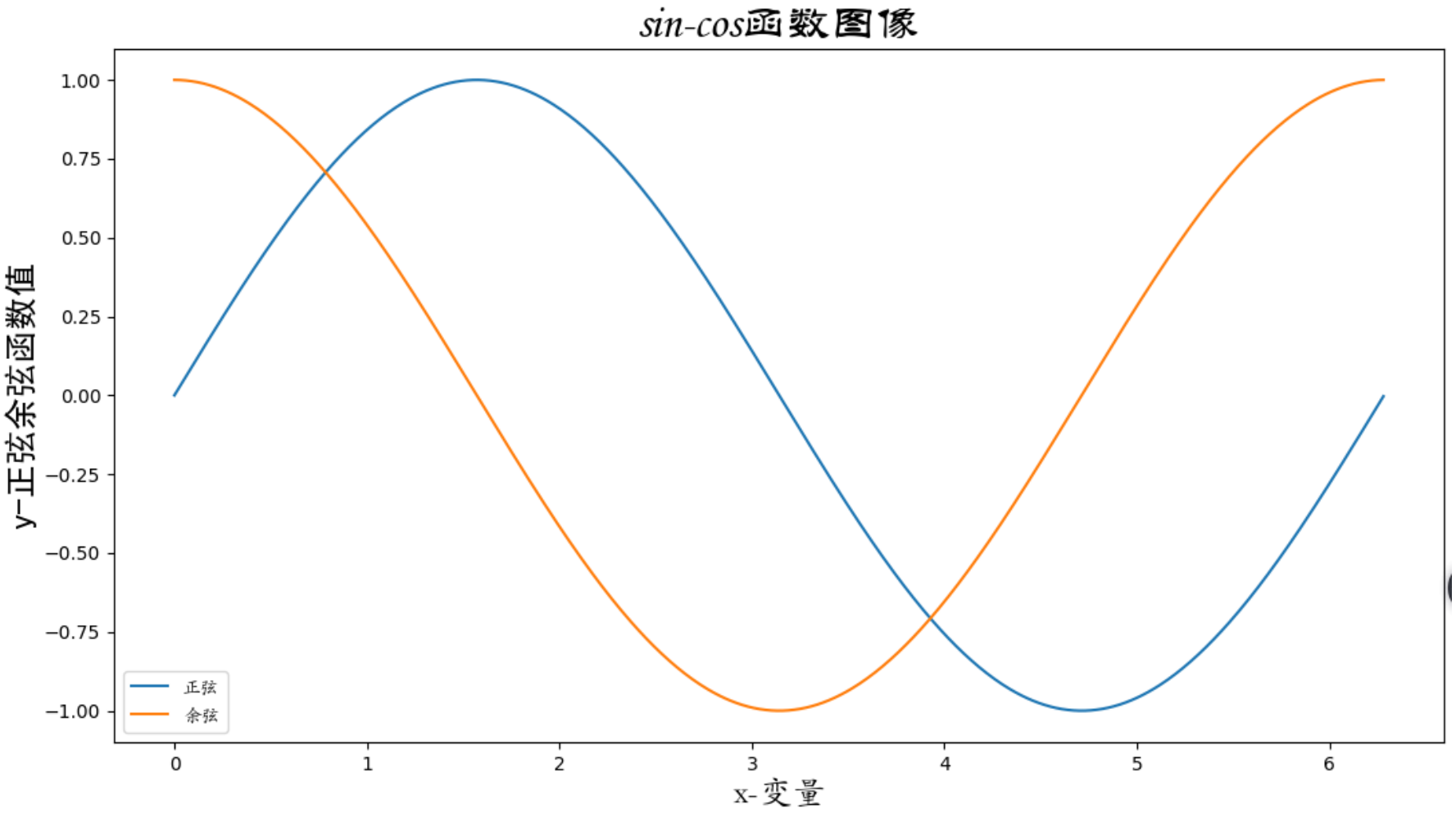
```
import numpy as np
import matplotlib.pylab as pl
import matplotlib.font_manager as fm
t = np.arange(0.0, 2.0*np.pi, 0.01)                    # 自变量取值范围
s = np.sin(t)                                          # 计算正弦函数值
z = np.cos(t)                                          # 计算余弦函数值
pl.plot(t, s, label='正弦')
pl.plot(t, z, label='余弦')
pl.xlabel( 'x-变量', fontproperties= 'STKAITI', fontsize=18) # 设置x标签，字体为华文楷体
pl.ylabel('y-正弦余弦函数值', fontproperties='simhei', fontsize=18)
pl.title('sin-cos函数图像', fontproperties='STLITI', fontsize=24)
#换一种方式设置字体
myfont = fm.FontProperties(fname=r 'C:\Windows\Fonts\STKAITI.ttf' )    #设置字体为华文楷体
pl.legend(prop=myfont)                                                  # 设置图例
pl.show()
```

- 所有的字体都存放在系统盘下的Fonts文件夹，系统盘下搜索*.ttf，即可找到该系统支持的所有字体文件。以下是各字体文件的对应关系

中文字体对应文件名

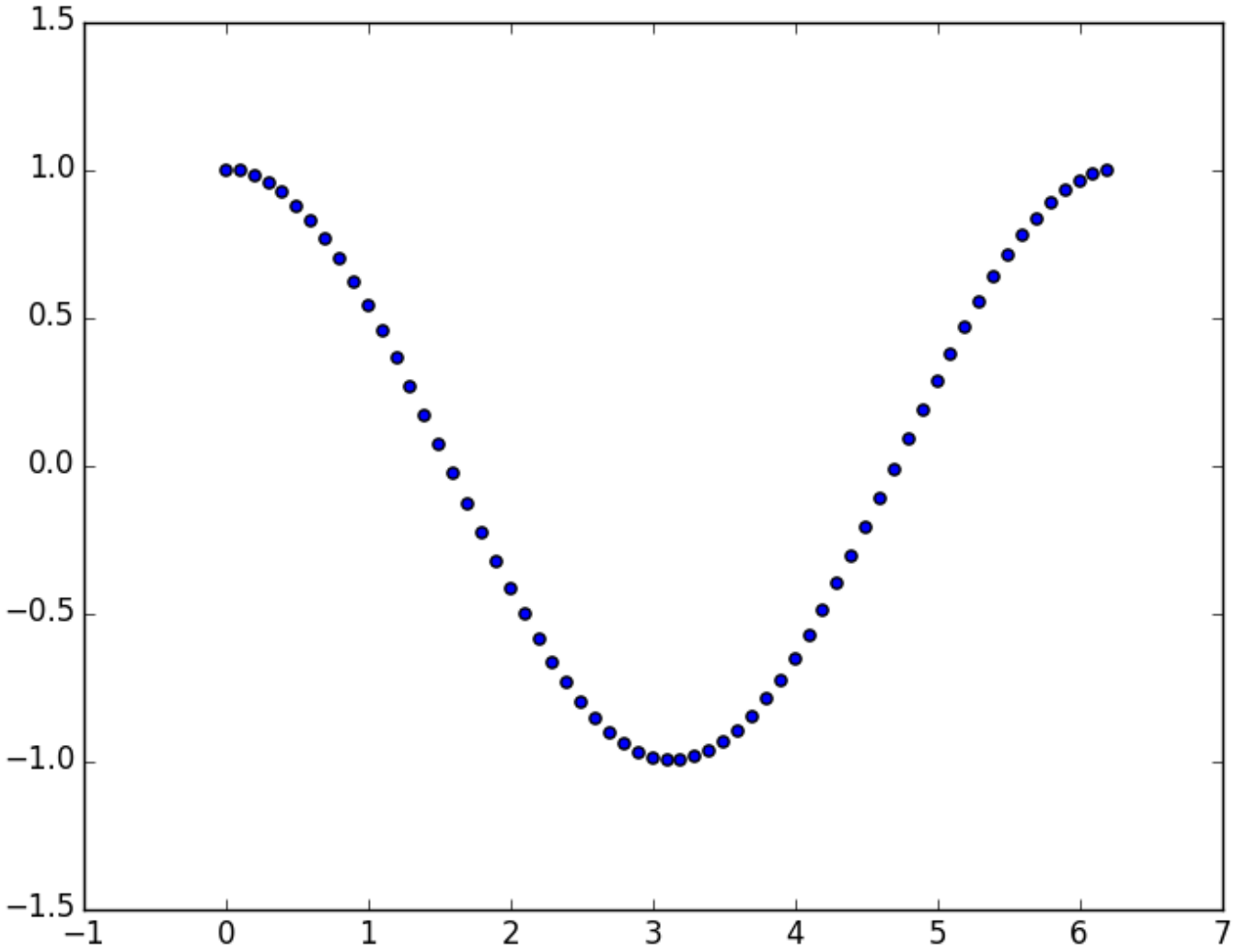
字体	文件名	字体	文件名
宋体	SIMSUN.TTF/simsunb.ttf	黑体	simhei.ttf
仿宋	simfang.ttf	楷体	simkai.ttf
方正舒体	FZSTK.TTF	方正姚体	FZYTK.TTF
隶书	SIMLI.TTF	华文彩云	STCAIYUN.TTF
华文细黑	STXIHEI.TTF	华文行楷	STXINGKAI.TTF
华文新魏	STXINWEI.TTF	华文中宋	STZHONGS.TTF
幼圆	SIMYOU.TTF	华文琥珀	STHUPO.TTF
华文楷体	STKAITI.TTF	华文隶书	STLITI.TTF
华文宋体	STSONG.TTF	新宋体	NSIMSUN.TTF

绘制带有中文标签和图例的图



绘制散点图

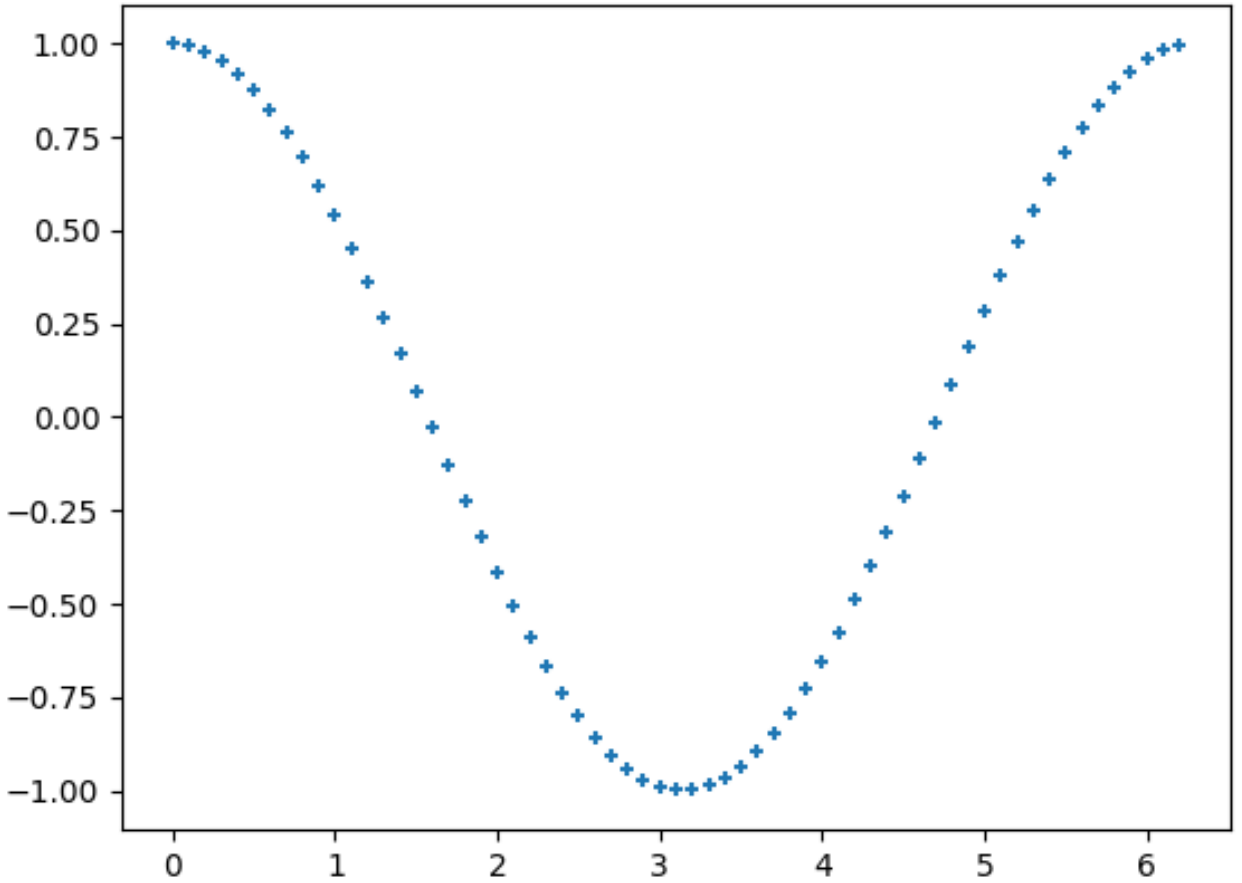
```
>>> a = np.arange(0, 2.0*np.pi, 0.1)
>>> b = np.cos(a)
>>> pl.scatter(a,b)
>>> pl.show()
```



绘制散点图

■修改散点符号与大小

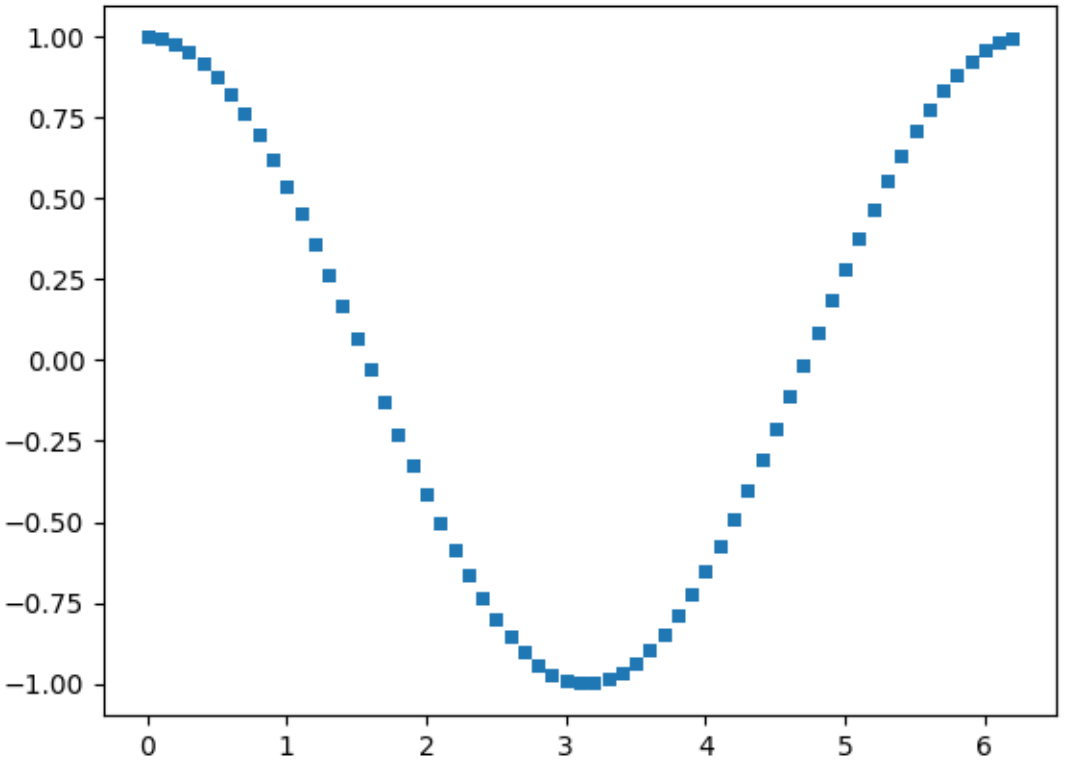
```
>>> pl.scatter(a,b,s=20,marker='+')
>>> pl.show()
```



绘制散点图

■修改线宽

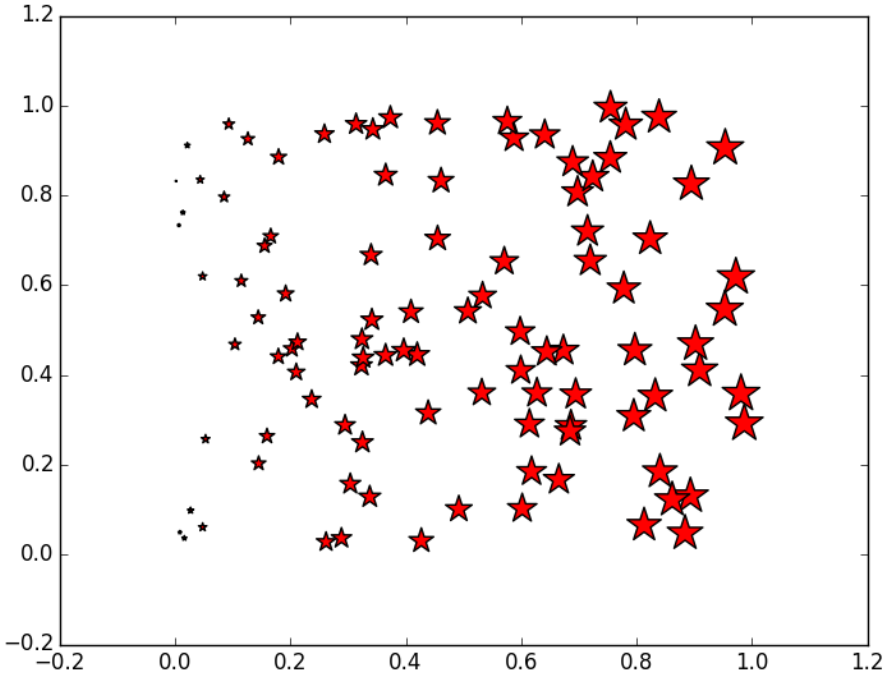
```
>>> pl.scatter(a,b,s=20,linewidths=5,marker= ‘+ ‘) #线太宽，显示更像菱形了
>>> pl.show()
```



绘制散点图

■修改颜色

```
>>> import matplotlib.pylab as pl
>>> import numpy as np
>>> x = np.random.random(100)
>>> y = np.random.random(100)
>>> pl.scatter(x,y,s=x*500,c=u'r',marker=u'*')
# s指大小，c指颜色，marker指符号形状
>>> pl.show()
```

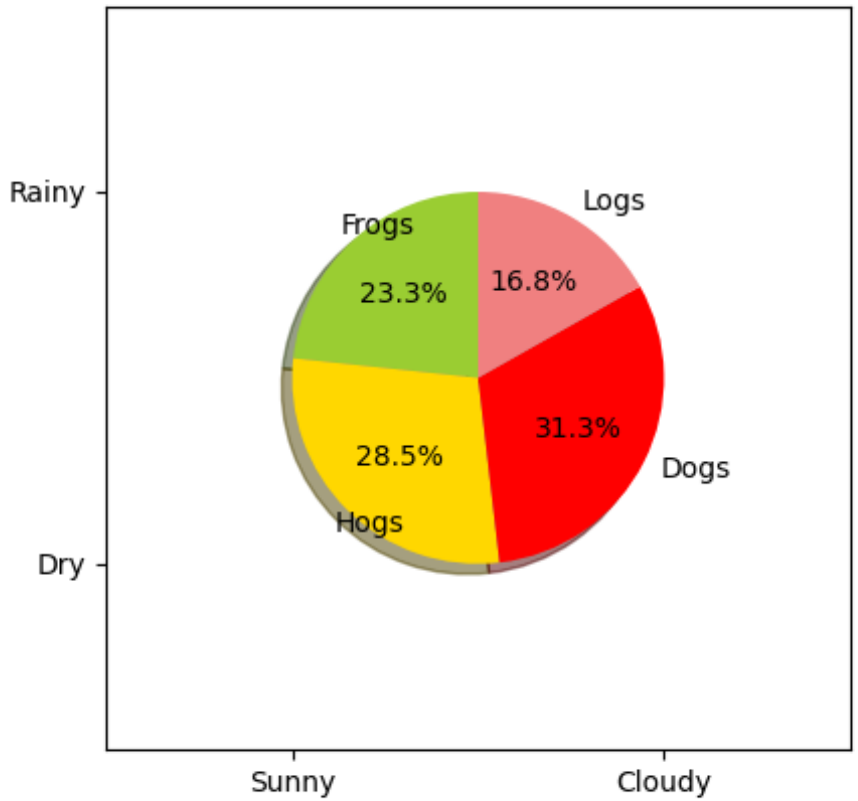


绘制饼状图

```
import numpy as np
import matplotlib.pyplot as plt


#The slices will be ordered and plotted counter-clockwise.
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
colors = ['yellowgreen', 'gold', '#FF0000', 'lightcoral']
explode = (0, 0, 0, 0)                    # 使饼状图中第2片和第4片裂开


fig = plt.figure()
ax = fig.gca()
```

10.1.3 绘制饼状图

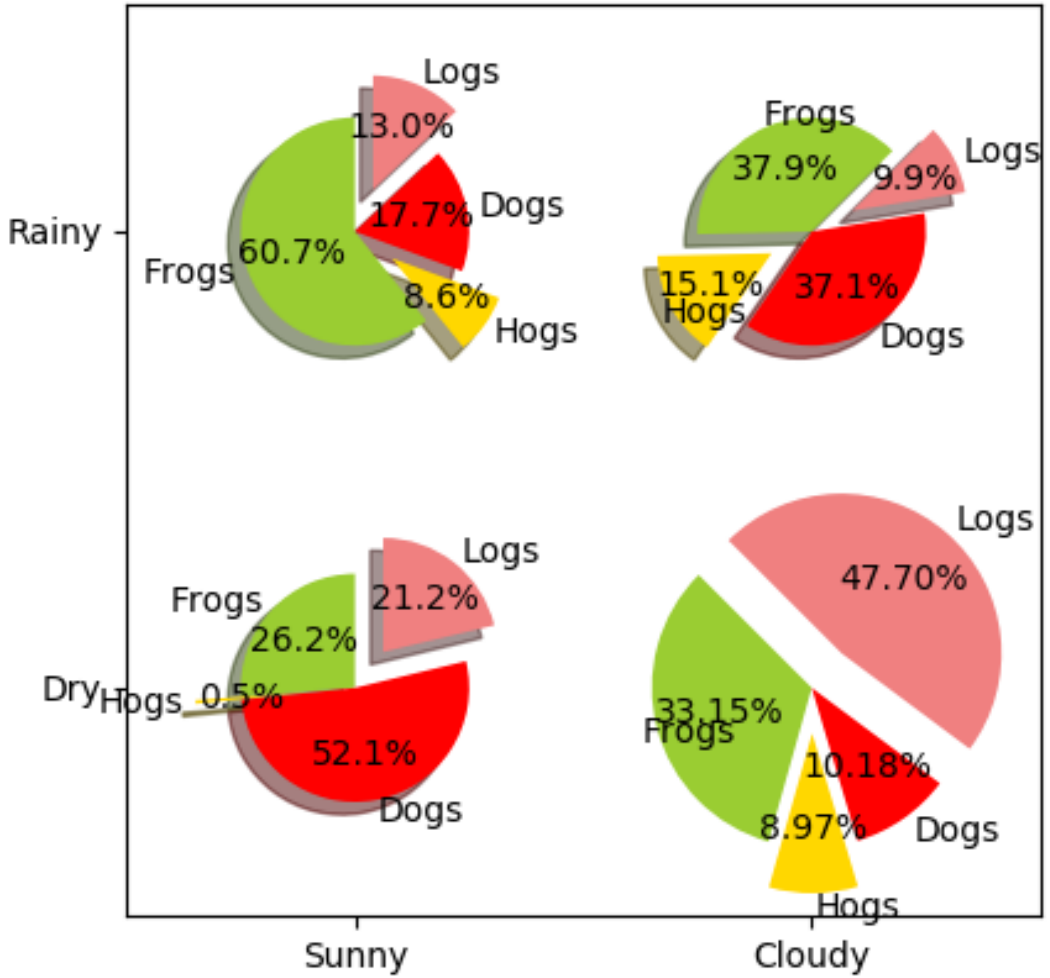
```
ax.pie(np.random.random(4), explode=explode, labels=labels, colors=colors,
       autopct='%1.1f%%', shadow=True, startangle=90,
       radius=0.5, center=(0.5, 0.5), frame=True)    # autopct设置饼内百分比的格式

ax.set_xticks([0, 1])                                # 设置坐标轴刻度
ax.set_yticks([0, 1])
ax.set_xticklabels(["Sunny", "Cloudy"])              # 设置坐标轴刻度上的标签
ax.set_yticklabels(["Dry", "Rainy"])
ax.set_xlim((-0.5, 1.5))                             # 设置坐标轴跨度
ax.set_ylim((-0.5, 1.5))
ax.set_aspect('equal')                               # 设置纵横比相等
plt.show()
```


Rainy
Dry
Sunny
Cloudy
Frogs
23.3%
Logs
16.8%
Hogs
28.5%
Dogs
31.3%

改变：饼图分裂，多图

```
explode = (0, 0.1, 0, 0.1)                    # 使饼状图中第2片和第4片裂开

ax.pie(np.random.random(4), explode=explode, labels=labels, colors=colors,
       autopct='%1.1f%%', shadow=True, startangle=90,
       radius=0.25, center=(0, 0), frame=True)    # autopct设置饼内百分比的格式
ax.pie(np.random.random(4), explode=explode, labels=labels, colors=colors,
       autopct='%1.1f%%', shadow=True, startangle=45,
       radius=0.25, center=(1, 1), frame=True)
ax.pie(np.random.random(4), explode=explode, labels=labels, colors=colors,
       autopct='%1.1f%%', shadow=True, startangle=90,
       radius=0.25, center=(0, 1), frame=True)
ax.pie(np.random.random(4), explode=explode, labels=labels, colors=colors,
       autopct='%1.2f%%', shadow=False, startangle=135,
       radius=0.35, center=(1, 0), frame=True)
```

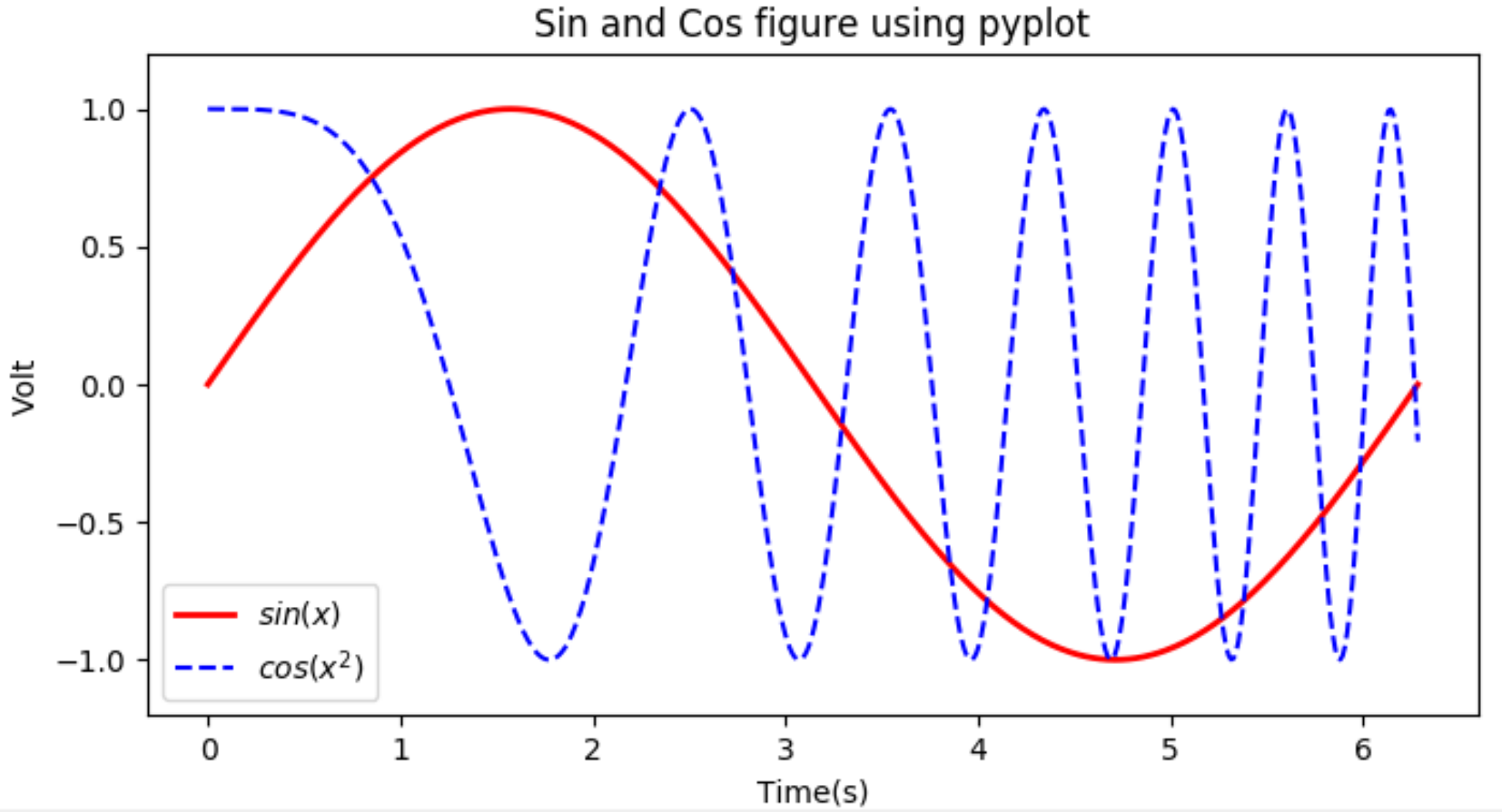
绘制饼状图


Rainy
Dry
Sunny
Cloudy
Logs
13.0%
Dogs
17.7%
Frogs
60.7%
Hogs
8.6%
Frogs
37.9%
Logs
9.9%
15.1%
Hogs
37.1%
Dogs
Logs
21.2%
Frogs
26.2%
Hogs
0.5%
52.1%
Dogs
Logs
47.70%
33.15%
Frogs
10.18%
8.97%
Dogs
Hogs

多个图形一起显示

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2*np.pi, 500)
y = np.sin(x)
z = np.cos(x*x)
plt.figure(figsize=(8,4))
# 标签前后加$将其显示为公式
plt.plot(x,y,label='$sin(x)$',color='red',linewidth=2)   # 红色，2个像素宽
plt.plot(x,z,'b--',label='$cos(x^2)$')                   # 蓝色，虚线
plt.xlabel('Time(s)')
plt.ylabel('Volt')
plt.title('Sin and Cos figure using pyplot')
plt.ylim(-1.2,1.2)
plt.legend()                                  # 显示图例
plt.show()                                    # 显示绘图窗口
```

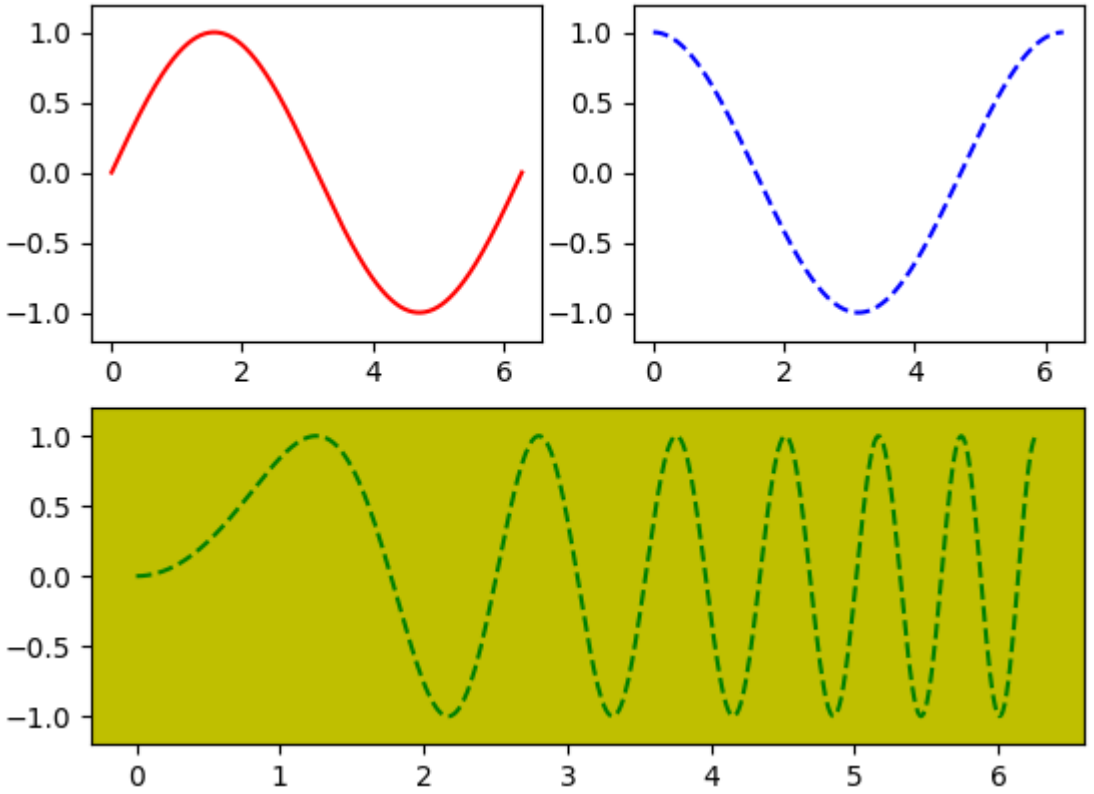
多个图形一起显示


Sin and Cos figure using pyplot
1.0
0.5
0.0
−0.5
−1.0
Volt
0
1
2
3
4
5
6
Time(s)
sin(x)
cos(x²)

多图单独显示

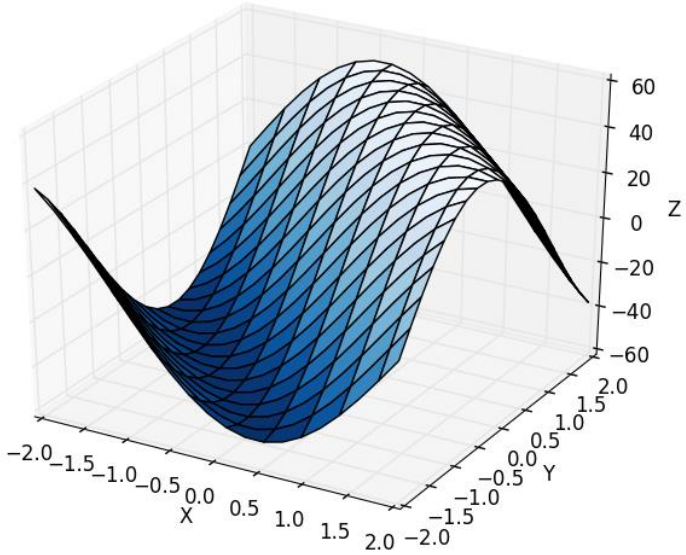
```
import numpy as np
import matplotlib.pyplot as plt

x= np.linspace(0, 2*np.pi, 500)          # 创建自变量数组
y1 = np.sin(x)                           # 创建函数值数组
y2 = np.cos(x)
y3 = np.sin(x*x)
plt.figure(1)                            # 创建图形
ax1 = plt.subplot(2,2,1)                 # 第一行第一列图形
ax2 = plt.subplot(2,2,2)                 # 第一行第二列图形
ax3 = plt.subplot(212, facecolor='y')    # 第二行
plt.sca(ax1)                             # 选择ax1
plt.plot(x,y1,color='red')               # 绘制红色曲线
plt.ylim(-1.2,1.2)                       # 限制y坐标轴范围
plt.sca(ax2)                             # 选择ax2
plt.plot(x,y2,'b--')                     # 绘制蓝色曲线
plt.ylim(-1.2,1.2)
plt.sca(ax3)                             # 选择ax3
plt.plot(x,y3,'g--')
plt.ylim(-1.2,1.2)
plt.show()
```

多图单独显示



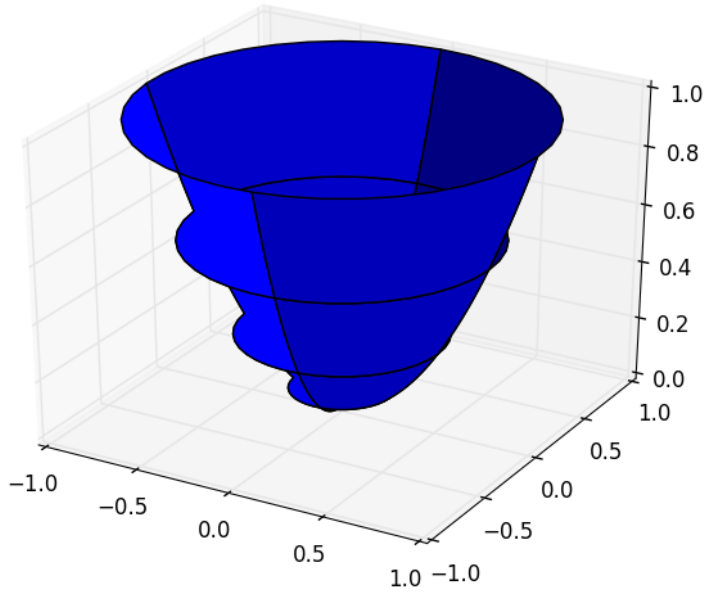
绘制三维图形

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d   # Matplotlib>mpl_toolkits>mplot3d>Axes3D


x,y = np.mgrid[-2:2:20j, -2:2:20j]       # 步长使用虚数
                                                    #  虚部表示点的个数
                                                    #  并且包含end

z = 50 * np.sin(x+y)                          # 测试数据
ax = plt.subplot(111, projection='3d')    # 三维图形
ax.plot_surface(x,y,z,rstride=2, cstride=1, cmap=plt.cm.Blues_r)
ax.set_xlabel('X')                             # 设置坐标轴标签
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```



绘制三维图形

```
import matplotlib.pylab as pl
import numpy as np
import mpl_toolkits.mplot3d
rho, theta = np.mgrid[0:1:40j, 0:2*np.pi:40j]
z = rho**2
x = rho*np.cos(theta)
y = rho*np.sin(theta)
ax = pl.subplot(111, projection='3d')
ax.plot_surface(x,y,z)
pl.show()
```



数据可视化 --相关标准库和扩展库

- Seaborn：在Matplotlib基础上提供了一个绘制统计图形的高级接口，为数据的可视化分析工作提供了极大的方便，使得绘图更加容易。

- Matplotlib绘图基本模仿MATLAB绘图库，其绘图风格和MATLAB类似。由于MATLAB绘图风格偏古典，因此，Python开源社区开发了Seaborn绘图模块，对Matplotlib进行封装，绘图效果更符合现代人的审美。

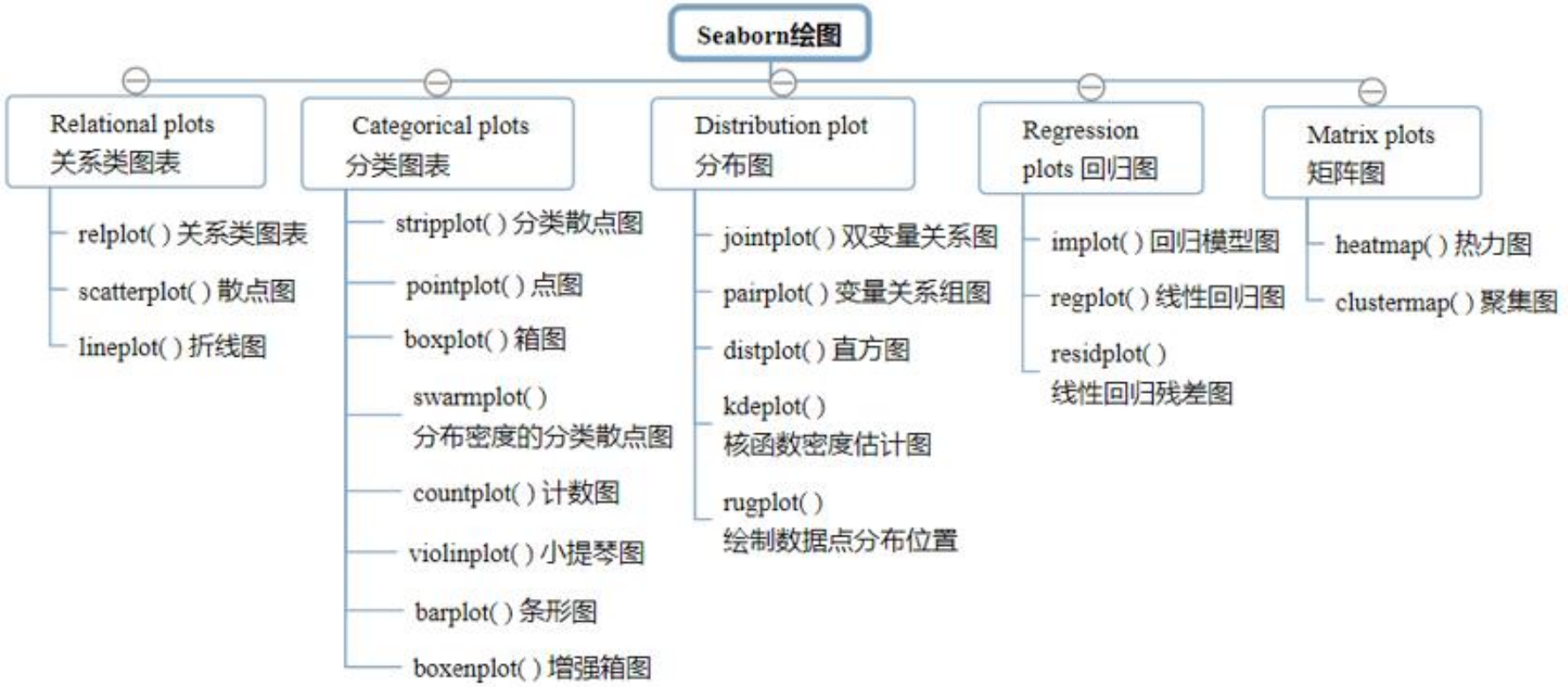
- Seaborn属于Matplotlib的一个高级接口，使得作图更加容易。在多数情况下使用Seaborn能做出很具吸引力的图，而使用Matplotlib可以制作具有更多特色的图。**应该把Seaborn视为Matplotlib的补充，而不是替代物**。

- 使用Seaborn时，使用的导入惯例为：

```
import seaborn as sns
```

seaborn简单应用

■Seaborn中共有5个大类21种绘图：

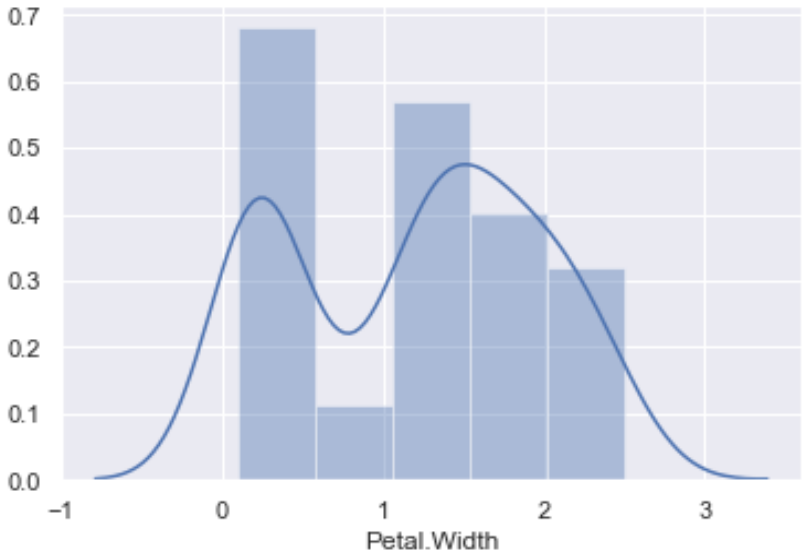


绘制直方图和密度曲线图

■Seaborn中利用distplot( )和kdeplot( )绘制直方图和密度曲线图，distplot( )为hist加强版，默认情况下绘制一个直方图，并嵌套一个对应的密度图。

■例：绘制iris数据集中Petal.Width的分布图

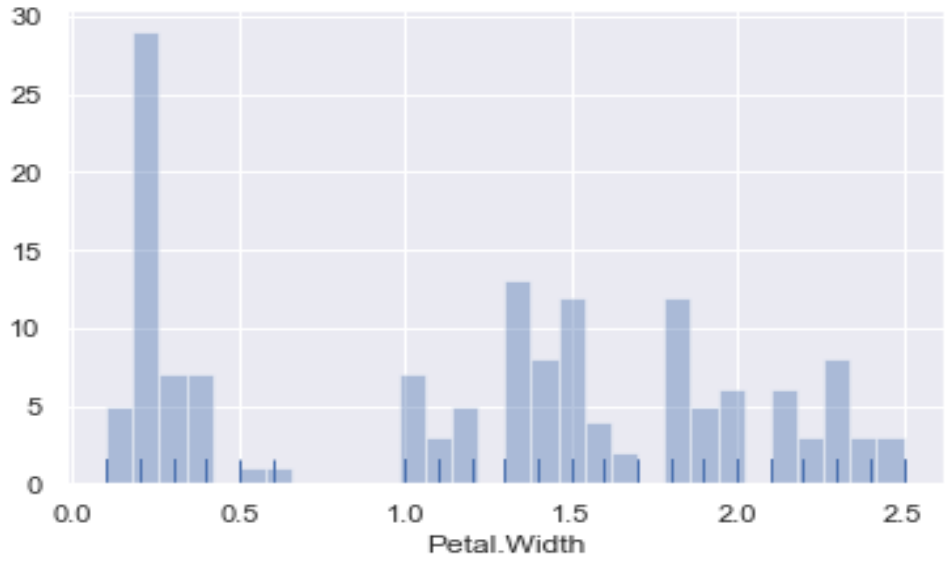
```
import matplotlib.pyplot as plt
df_iris=pd.read_csv('iris.csv')
sns.set(color_codes=True)
sns.distplot(df_iris['Petal.Width'])
```



绘制直方图和密度曲线图

使用distplot方法绘制的直方图与matplotlib是类似的。在distplot的参数中，可以选择不绘制密度图。其中的rug参数绘制毛毯图，可以为每个观测值绘制小细线（边际毛毯），也可以单独用rugplot进行绘制。

```
sns.distplot(df_iris['Petal.Width'],bins=30,kde=False,rug=True)
```

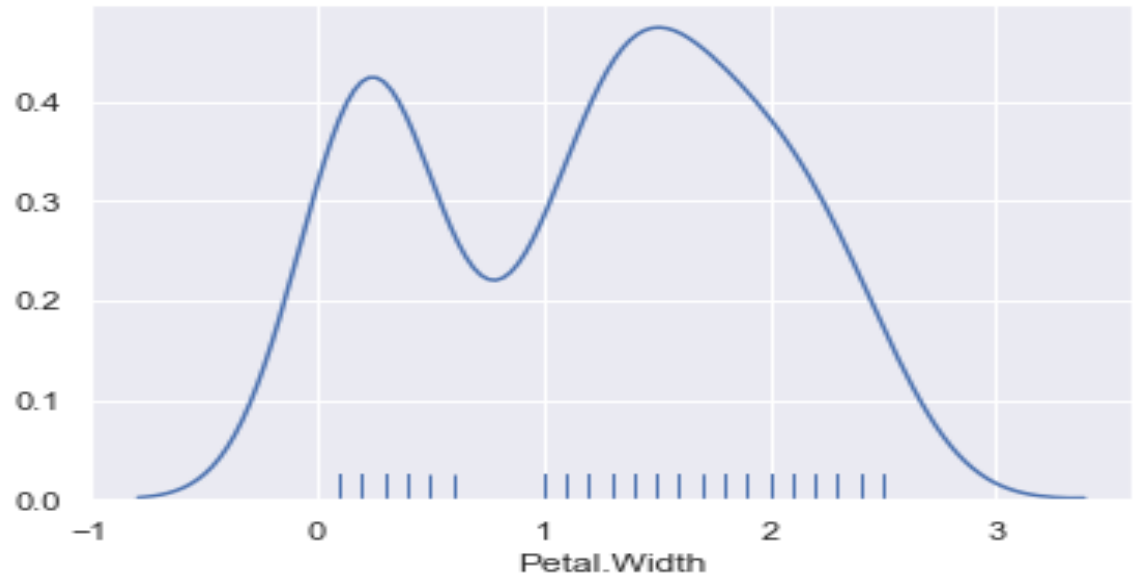


A marginal rug plot is essentially a one-dimensional scatter plot that can be used to visualize the distribution of data on each axis

绘制直方图和密度曲线图

■ 如果设置hist为False，则可以直接绘制密度图而没有直方图。

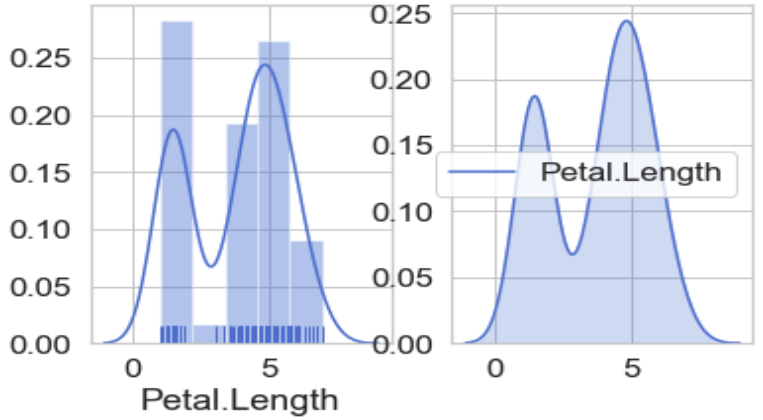
例：sns.distplot(df_iris['Petal.Width'],hist=False,rug=True)



绘制直方图和密度曲线图

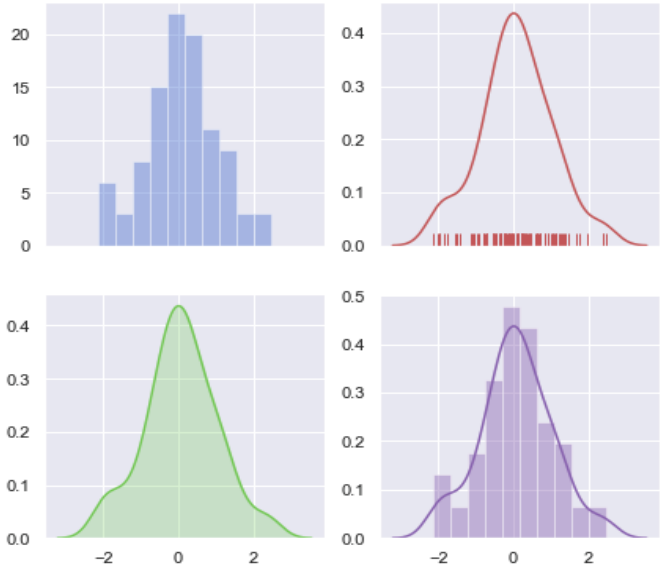
- 利用distplot函数可以同时绘制直方图、密度图和毛毯图，同时，这些分布图都有对应的专门函数。其中，kdeplot函数绘制密度图，rugplot用于绘制毛毯图。

```
import matplotlib.pyplot as plt
df_iris=pd.read_csv('iris.csv')
fig,axes=plt.subplots(1,2)
sns.distplot(df_iris['Petal.Length'],ax=axes[0
],kde=True,rug=True) #kde密度曲线，rug边际毛毯
sns.kdeplot(df_iris['Petal.Length'],ax=axes[1]
,shade=True)  #shade阴影
plt.show()
```



绘制直方图和密度曲线图

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
rs=np.random.RandomState(10)
d=rs.normal(size=100)
f,axes=plt.subplots(2, 2, figsize=(7, 7), sharex=True)
sns.distplot(d, kde=False, color="b", ax=axes[0,0])
sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0,1])
sns.distplot(d, hist=False,color="g", kde_kws={"shade":True},
ax=axes[1,0])
sns.distplot(d, color="m", ax=axes[1,1])
plt.show()
```

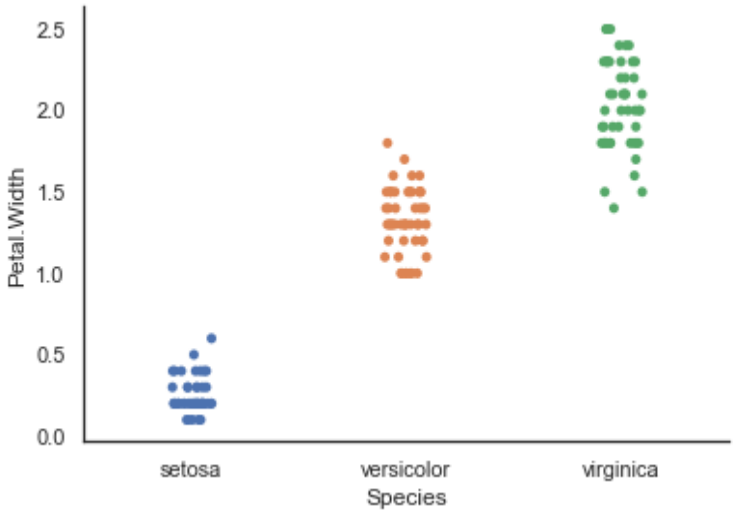


绘制散点图

■在Seaborn中，利用stripplot绘制各变量在每个类别的值。

例：在iris数据集中，显示Petal.Width在Species上值的分布

```
sns.set(style='white',color_codes=True) #设置样式
sns.stripplot(x=df_iris['Species'],y= df_iris['Petal.Width'],data=df_iris)
sns.despine()    #去坐标轴
```

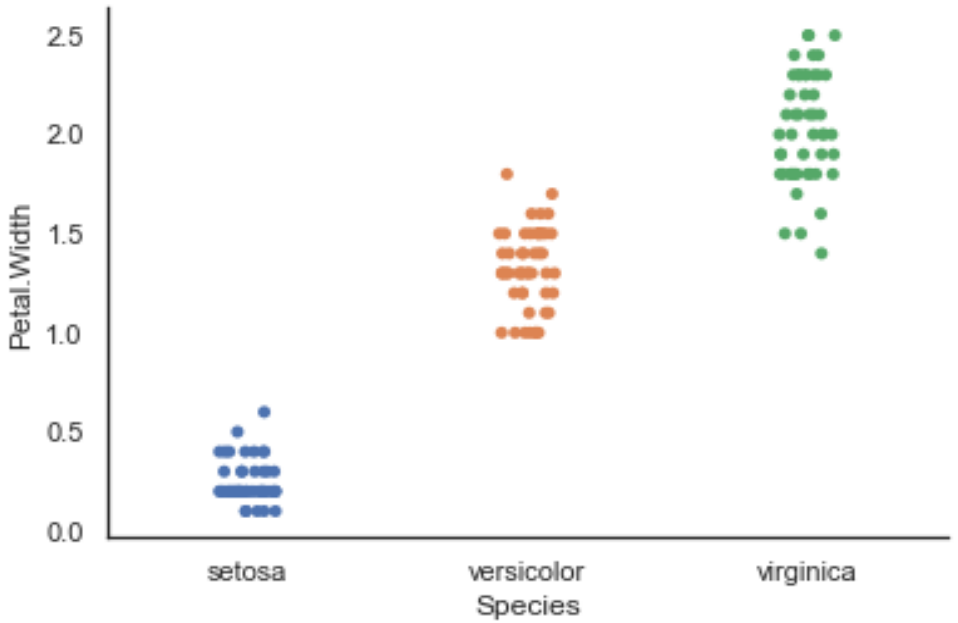


10.2.2 绘制散点图

■由于散点图中数据众多，很多点会被覆盖，这时可以加入抖动（jitter=True）。

例：

```
sns.stripplot(x=df_iris['Species'],y= df_iris['Petal.Width'],data=df_iris,jitter=True)
sns.despine()   #去坐标轴
```

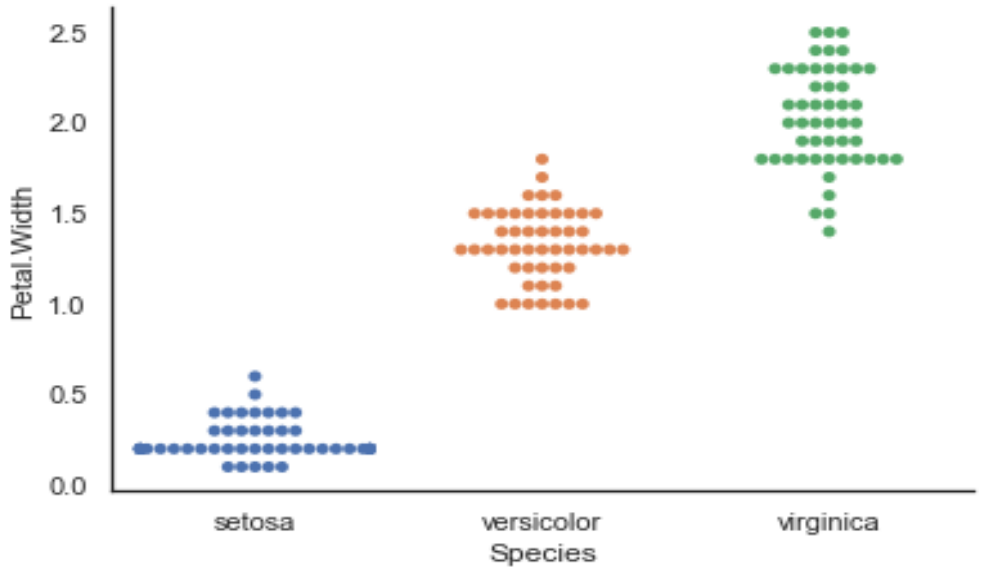


绘制散点图

■如果需要看清每个数据点，可以使用swarmplot函数。

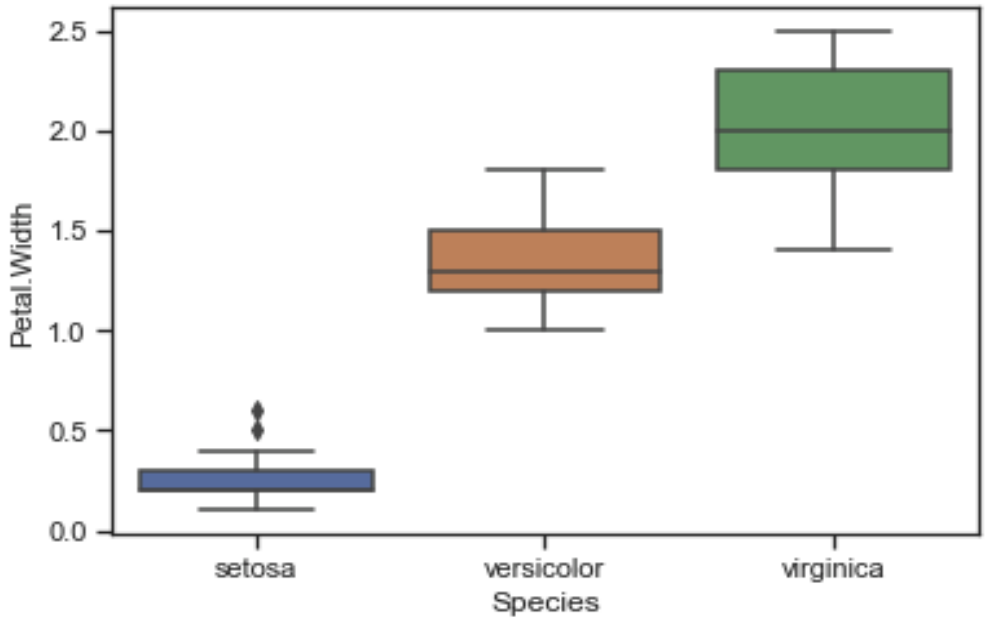
例：

```
sns.swarmplot(x=df_iris['Species'],y= df_iris['Petal.Width'],data=df_iris)
sns.despine()   #去坐标轴
```



绘制箱线图

```
df_iris=pd.read_csv('iris.csv')
sns.set(style="ticks")
sns.boxplot(x=df_iris['Species'],y = df_iris['Petal.Width'])
plt.show()
```



变量相关性

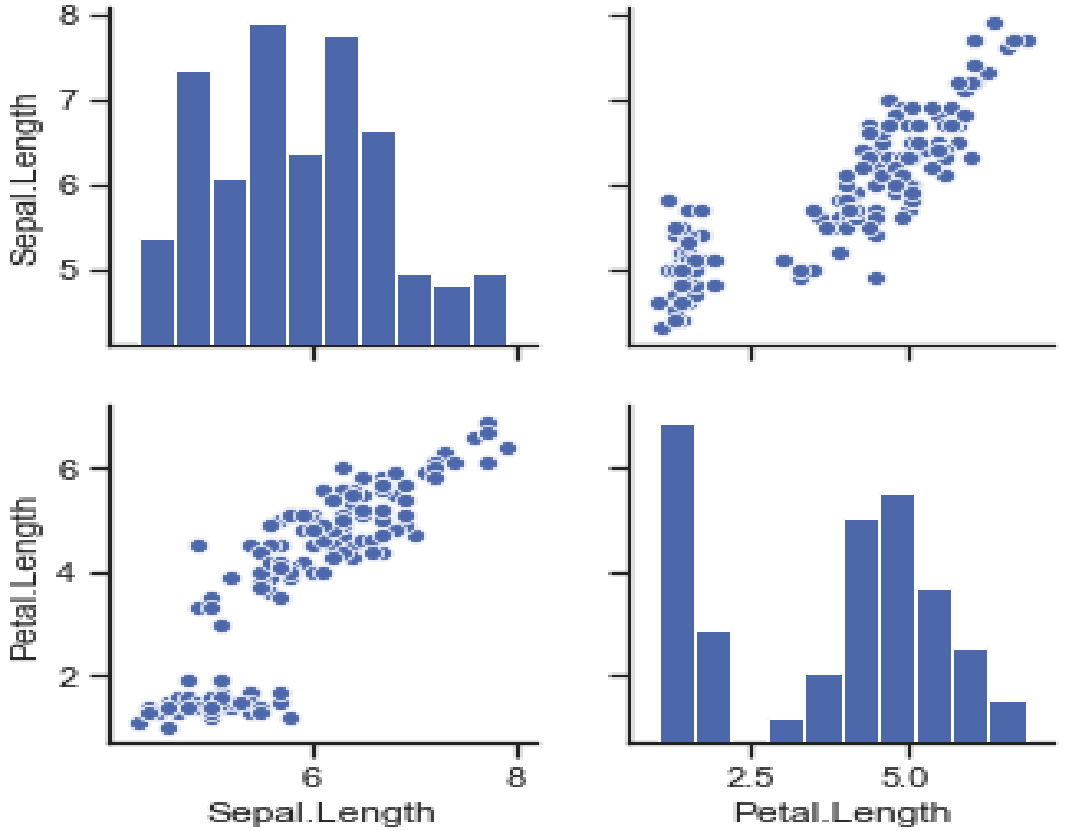
- 在seaborn中利用 pairplot（）实现数据特征的两两对比。默认是所有特征，可以通过vars参数指定部分特征。

```
seaborn.pairplot(data, hue=None, hue_order=None, palette=None, vars=Non
e, x_vars=None, y_vars=None, kind='scatter', diag_kind='auto', markers=
None, height=2.5, aspect=1, dropna=True, plot_kws=None, diag_kws=None,
grid_kws=None, size=None)
```

- pairplot主要展现的是**变量两两之间的关系**（线性或非线性，有无较为明显的相关关系）

变量相关性

```
df_iris=pd.read_csv('iris.csv')
sns.set(style="ticks")
g = sns.pairplot(df_iris,vars=['Sepal.Length', 'Petal.Length'])
```

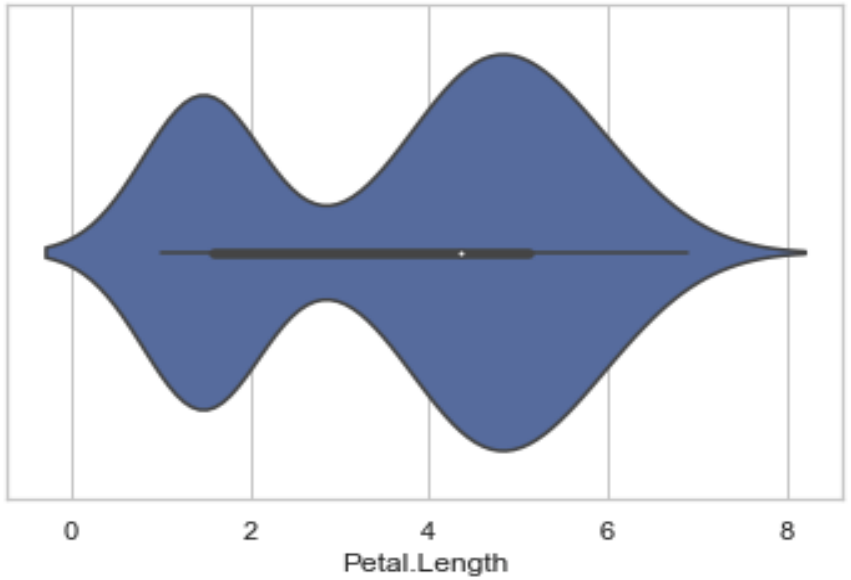


10.2.5 绘制小提琴图

- 小提琴图其实是箱线图与核密度图的结合，箱线图展示了分位数的位置，小提琴图则展示了任意位置的密度，通过小提琴图可以知道哪些位置的密度较高。在图中，白点是中位数，黑色盒型的范围是下四分位点到上四分位点，细黑线表示须。外部形状即为核密度估计（在概率论中用来估计未知的密度函数，属于非参数检验方法之一）。

绘制小提琴图

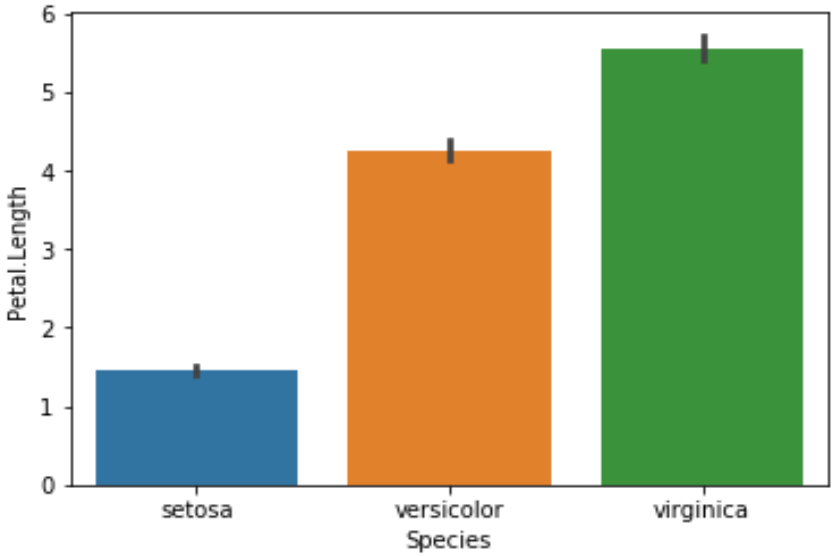
```
sns.set_style("whitegrid")
df_iris=pd.read_csv('iris.csv')
ax = sns.violinplot(x=df_iris['Petal.Length'])
```



10.2.5 绘制柱状图

- 在Seaborn中使用barplot函数绘制柱状图，默认情况下，绘制的y轴是平均值

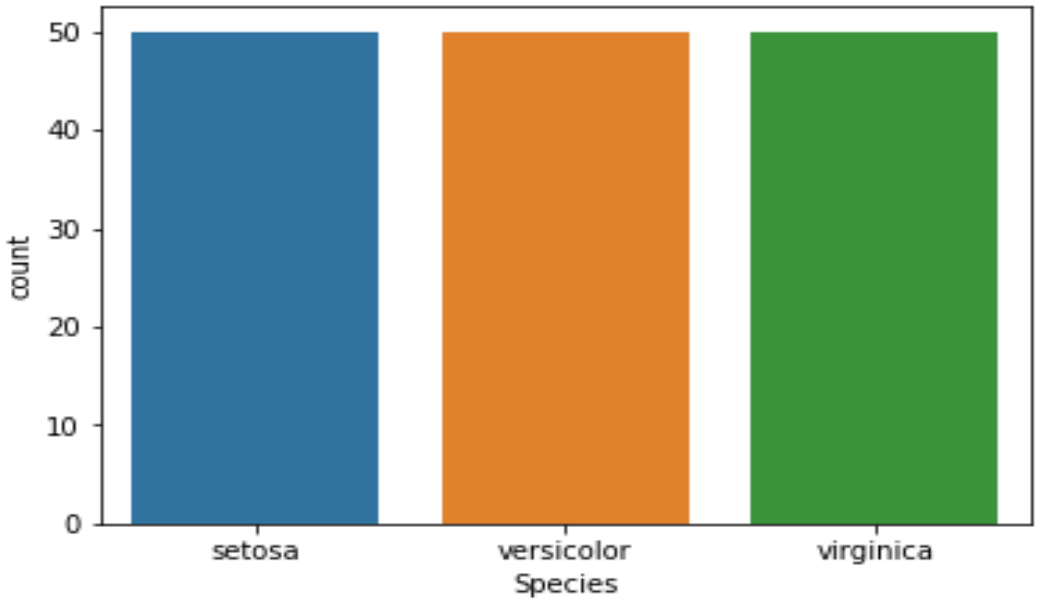
```
df_iris=pd.read_csv('iris.csv')
sns.barplot(x=df_iris['Species'],y=df_iris['Petal.Le
ngth'],data=df_iris)
```



10.2.5 绘制柱状图

■在柱状图中，经常会绘制类别的计数柱状图，在matplotlib中需要对DataFrame进行计算，而在Seaborn中则使用countplot函数即可。

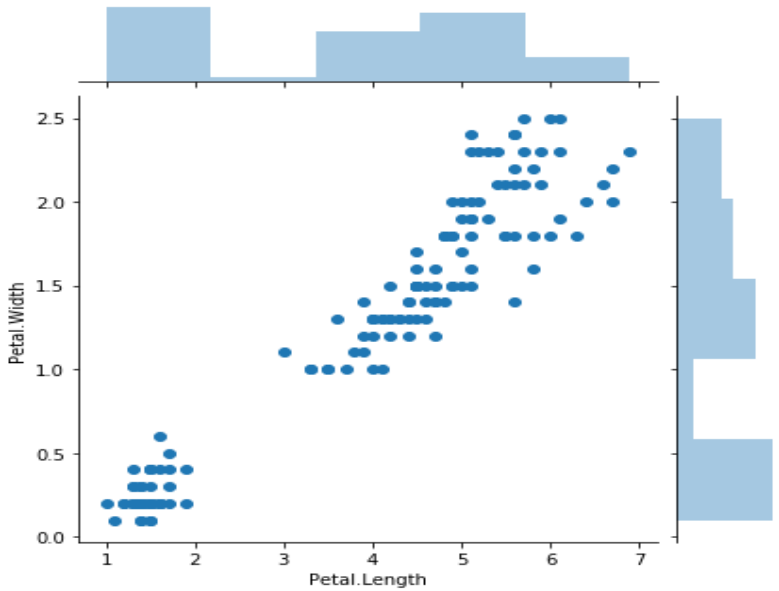
```
sns.countplot(x='Species',data=df_iris)
```



10.2.6 绘制多变量图

- 在matplotlib中，为了绘制两个变量的分布关系，常使用散点图的方法。在Seaborn中，使用jointplot函数绘制一个多面板图，不仅可以显示两个变量的关系，还可以显示每个单变量的分布情况。

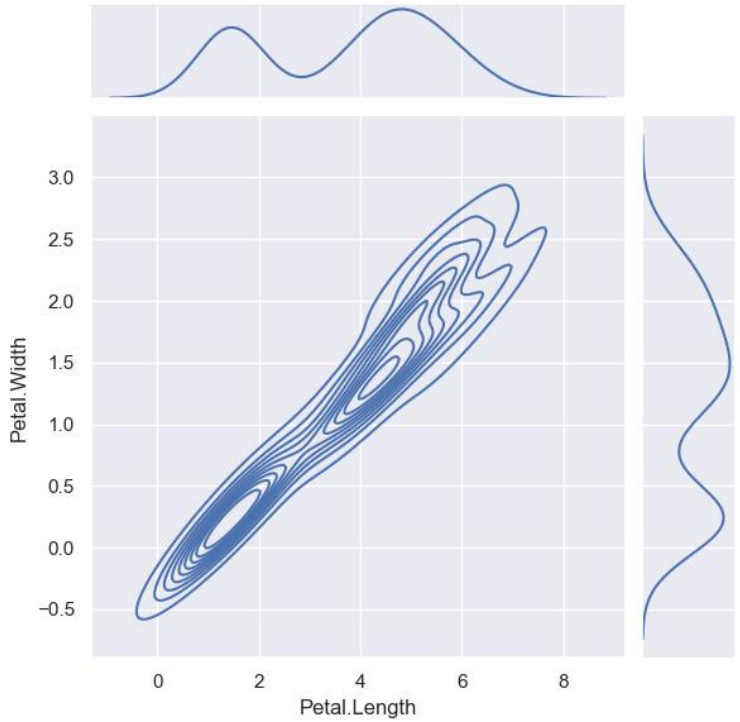
```
sns.jointplot(x='Petal.Length',y='Petal.Width' ,data=df_iris)
```



10.2.6 绘制多变量图

■在jointplot函数中，改变kind参数为kde，但变量的分布就用密度图来代替，而散点图则会被等高线图代替。

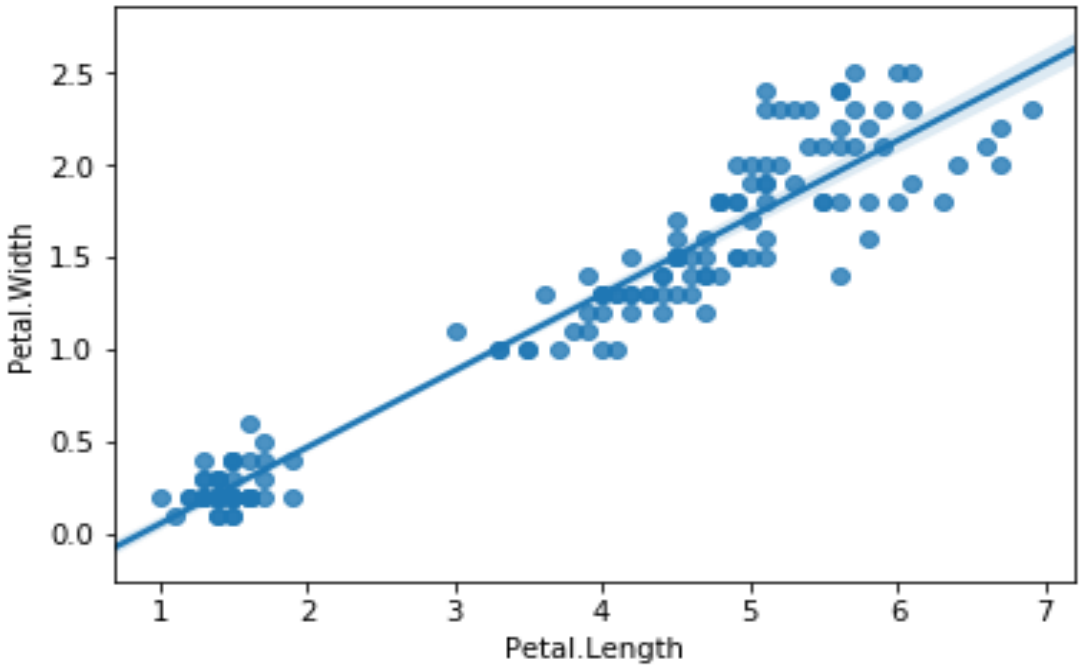
```
sns.jointplot(x='Petal.Length',y='Petal.Width'
,data=df_iris,kind='kde')
```



10.2.7 绘制回归图

■绘制回归图可以揭示两个变量间的线性关系。Seaborn中，使用regplot函数绘制回归图。

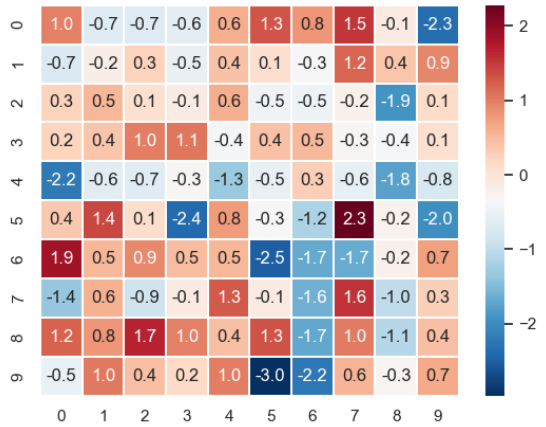
```
sns.regplot(x='Petal.Length',y='Petal.Width' ,data=df_iris)
```



10.2.8 绘制热力图

- 热力图是数据可视化项目中比较常用的数据显示方式。热力图通过颜色变化程度直观反应出热点分布，区域聚集等数据信息。热力图实现过程就是通过简单的数学变化，将离散的点信息映射为图像。

```
data_new = np.random.randn(10,10)

#热力图主要展示的是二维数据的数据关系, 不同大小的值对应不同的颜色深浅
sns.heatmap(data_new)

#cmap：设置颜色带的色系
#sns.heatmap(data_new,cmap="RdBu_r")

#annot：是否显示数值注释
#sns.heatmap(data_new,cmap="RdBu_r",annot=True)

#fmt：format的缩写，设置数值的格式化形式;linewidths：控制每个小方格之间的间距
#sns.heatmap(data_new,cmap="RdBu_r",annot=True,fmt="1.1f",linewidths=0.3)

plt.show()
```



提交次数：26，共 26 次

你对编程语言（如C++/Java/Python）的掌握程度

熟练精通（打过信息竞赛或开发过程序）	1 位答题者	4 %
完全没学过	17 位答题者	**65 %** ✓
略懂略懂（自学或者兴趣班）	6 位答题者	23 %
认真学过（比如1个假期或学期）	2 位答题者	8 %

你对编程中的数据结构和算法是否了解

完整了解过，有点难啊	3 位答题者	12 %
精通，非常有感觉		0 %
知道它是干嘛的	7 位答题者	27 %
你说啥?	16 位答题者	**62 %** ✓

你学完这门课后

还没想过这问题	2 位答题者	8 %
很想应用，但老师教得不好就算了	6 位答题者	23 %
学习嘛，打怪过关而已		0 %
也许有机会就试试?	18 位答题者	69 %

你对这门课的期望

随便听听，有备无患		0 %
飘过~~~~~无期望		0 %
正有此意，掌握一门手艺	19 位答题者	73 %
以后可能用得上，需要事先了解些	7 位答题者	27 %

你想象这门课是

那是相当地难懂	4 位答题者	15 %
总是能过滴	3 位答题者	12 %
吃不太准，但感觉不会轻松	19 位答题者	73 %
小case啦		0 %

如果课程过程中有小组讨论和作业，你会

看其他人的参与程度呗	15 位答题者	58 %
我的想法如滔滔江水，我的实力强大无比，就等着这个机会	4 位答题者	15 %
当然是引领潮流啦	5 位答题者	19 %
我很忙的	2 位答题者	8 %

回顾：微调

- 几乎无基础
  - 主讲基础，耐心细致，控制难度
- 应用期望高
  - 多些实践案例和应用
- 自信和积极性弱
  - 前期放慢节奏，加强课堂互动

商学院学生学习信息技术（编程）的意义

- 完整的数据团队
  - 数据管理（数据库）
  - + 数据处理（程序设计/业务逻辑）
  - + 数据分析（算法/统计）
  - + 数据产品（程序设计/开发）

以社会化媒体数据（social media）为例

- 数据管理
  - 数据获得
    - 网络编程(requests)
  - 数据存储
    - 结构（list/dic, numpy/array)
    - 文件（file, panda）
- 数据处理
  - 逻辑结构
    - 模块化（函数、类）
    - 循环判断 (for/while/if)
    - 异常(exception)
  - 文本处理
    - 文本字符（string）
    - 正则(reg)
- 数据分析
  - 科学计算(numpy, scipy)
  - 数据展示(matplotlib)

商学院学生学习信息技术（编程）的意义

以社会化媒体数据（social media）为例

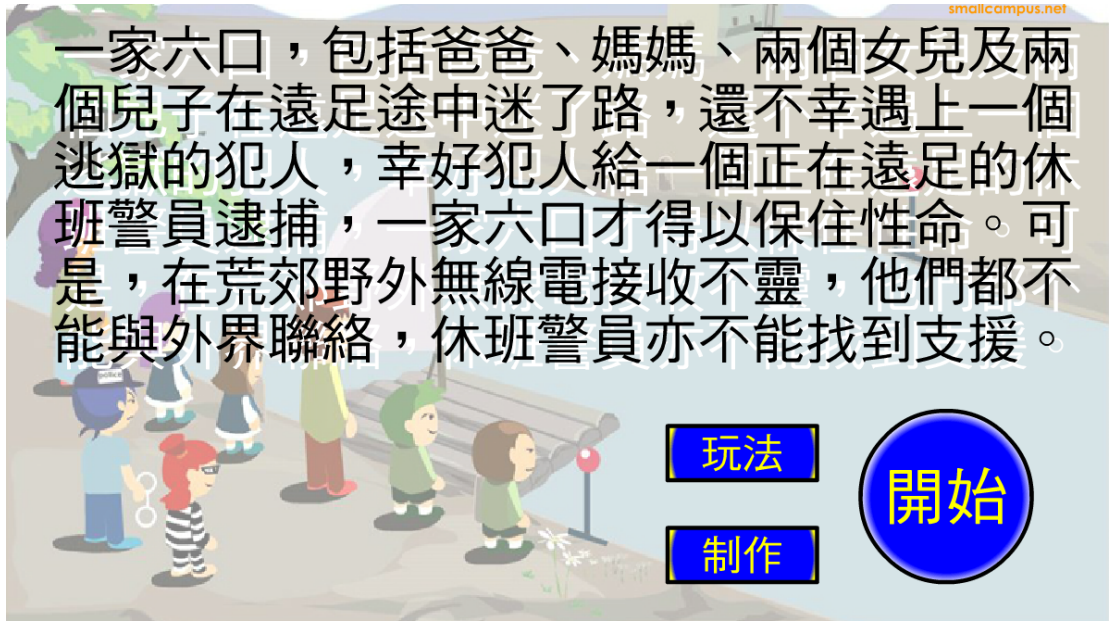
- 数据管理
  - 数据获得
    - 网络编程(requests)
  - 数据存储
    - 结构（list/dic, numpy/array)
    - 文件（file, panda）
- 数据处理
  - 逻辑结构
    - 模块化（函数、类）
    - 循环判断 (for/while/if)
    - 异常(exception)
  - 文本处理
    - 文本字符（string）
    - 正则(reg)
- 数据分析
  - 科学计算(numpy, scipy)
  - 数据展示(matplotlib)

未来进阶

- 数据管理
  - 数据库
  - 数据结构与算法
- 数据处理
  - 自然语言/多媒体数据处理
- 数据分析
  - 人工智能/机器学习算法（多种）
  - 数据可视化
  - 大数据分析
  - 社会网络分析
  - 统计回归/计量分析
- *数据产品/系统
  - 系统设计与开发

到期末，我们可以做到什么程度？

8人过河一血



8人过河问题

- 抽象现实问题
  - 设A为河一边的状态，用集合{A}分别表示父亲，母亲；儿子1；儿子2；女儿1；女儿2；警察；小偷的状态。=1 表示人在，=0表示人不在。
    - 则初始A=(1,1,1,1,1,1,1,1)
  - 设B为河另外一边的状态数组，其含义与A数组相同。
    - 则初始B=(0,0,0,0,0,0,0,0)。
  - 完成过河任务后，A=(0,0,0,0,0,0,0,0); B=(1,1,1,1,1,1,1,1)

- left, right, ship=['警察','犯人','父亲','儿子','儿子','母亲','女儿','女儿'],[],[]

- #操作两岸和船的状态
- ship.append(), left.pop(), right.append()

- #假设i，j是记录每次登船的人
- ship.append(left.pop(i))
- ship.append(left.pop(j))

- #如果满足两岸和划船的条件
- for k in ship:
- 	right.append(k)

- #i为list，比如left, right,ship

- if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
- 　　……
- if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
- 　　……
- if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
- 　　……

```
x=[left,right,ship]
    for i in x:
        if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
            ……
        if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
            ……
        if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
            ……
        if (i==ship):
            if '父亲' not in i and '母亲' not in i and '警察' not in i:
                ……
```

到此，大家觉得还差啥？

```
#定义场景结构
left, right, ship=['警察','犯人','父亲','儿子','儿子','母亲','女儿','女儿'],[],[]

#操作两岸和船的状态
ship.append(), left.pop(), right.append()

#假设i，j是记录每次登船的人
ship.append(left.pop(i))
ship.append(left.pop(j))

#如果满足两岸和划船的条件
 for k in ship:
    right.append(k)
```

```
# 遍历状态，确认符合规则
x=[left,right,ship]
   for i in x:
      if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
         ……
      if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
         ……
      if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
          ……
      if (i==ship):
         if '父亲' not in i and '母亲' not in i and '警察' not in i:
            ……
```

首先：定义世界

```
# 定义世界：左岸、右岸、船和人物
left,ship,right=['警察','犯人','父亲','儿子','儿子','母亲','女儿','女儿'],[],[]


# 需要设定一些临时变量及赋予其初始值的话（比如用于判断循环什么时候结束，索引，备份初始队列信息等）
# 定义规则
def rules(left1,right1,ship1):
    x=[left1,right1,ship1]
    for i in x:
        if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
            # 标识不允许
            return False
        if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
            # 标识不允许
            return False
        if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
            # 标识不允许
            return False
        if (i==ship1):
            if '父亲' not in i and '母亲' not in i and '警察' not in i:
                # 标识不允许
                return False
```

其次，定义行动

```
# 定义左行动函数，考虑船从左到右的情况
def actions_1(ship1,right1,left1):
    # 通常需要考虑涉及的全局/局部变量，以及临时变量（比如索引i,j)
    # 无限循环直到不触碰规则
    while True:
        # 从left列表选出两个元素移到ship中
        ship1.append(left1.pop(i))
        ship1.append(left1.pop(j)))
        # 使用for循环遍历ship中的元素移到right列表中
        for k in ship1:
            right1.append(k))
        # 注意，先不删除船中的元素，判断本次运行过程是否触发规则
        # 判断左岸、船上、右岸的人员是否都会安全，不安全则报错
        try:
            rules(left1,right1,ship1))
```

```
# 注意，还需要考虑从右到左的船上和从左到右的船上两人是否一样，若是一样的会陷入循环，报错
            if sorted(ship1)==sorted(ship2):
                return False
        # 若都不报错，则行动正确，结束循环
            break
        # 若try中报错，则调整索引进行后续变化/遍历（尝试自己填下）
        except:
# 结束时清理首尾，比如调整各种队列状态和索引
```

其次，定义行动

```
# 定义右行动函数，考虑船从右到左；和上面的不同主要在于优先找到1个大人回来
def actions_2(ship1,right1,left1):
    # 无限循环直到不触碰规则
    while True:
        # 从right列表选出一个元素移到ship中（优先挪1个人划船）
        ship1.append(right1.pop(t))
        # 使用for循环遍历ship1中的元素移到left1列表
        # 注意：先不删除船中的元素，用以判断本次运行过程是否触发规则）
            # 判断左岸、船上、右岸的人员是否都会安全，不安全则报错
            # 结束循环（这里一个人和两个人不可能重复，不用判断是否与上次船相等导致的循环）
            # 若try中报错，则调整索引进行变化
            # 超出列表则结束循环

    # 如果需要从右边运输两个人到左边，类似action_1
```

```
# 主程序（通过调用rule, action函数，完成遍历）

# 无限循环直到运输完毕
# 引用左右船两个函数并输出左岸的人数，以及船上的人
while True:
    actions_1(ship,right,left)
    print(f'{ship2}从左往右')
    print(f'此时左边剩下{left}，右边剩下{right}')
    actions_2(ship,right,left)
    print(f'{ship2}从右往左')
    print(f'此时左边剩下{left}，右边剩下{right}')

# 由于最后一段是从左岸到右岸，判断左岸人数是否为2，是则进行最后一次运输并结束循环
    if len(left)==2:
        actions_1(ship,right,left)
        print(f'{ship2}从左往右')
        print(f'此时左边剩下{left}，右边剩下{right}')
        break
```

到期末，我们可以做到什么程度？

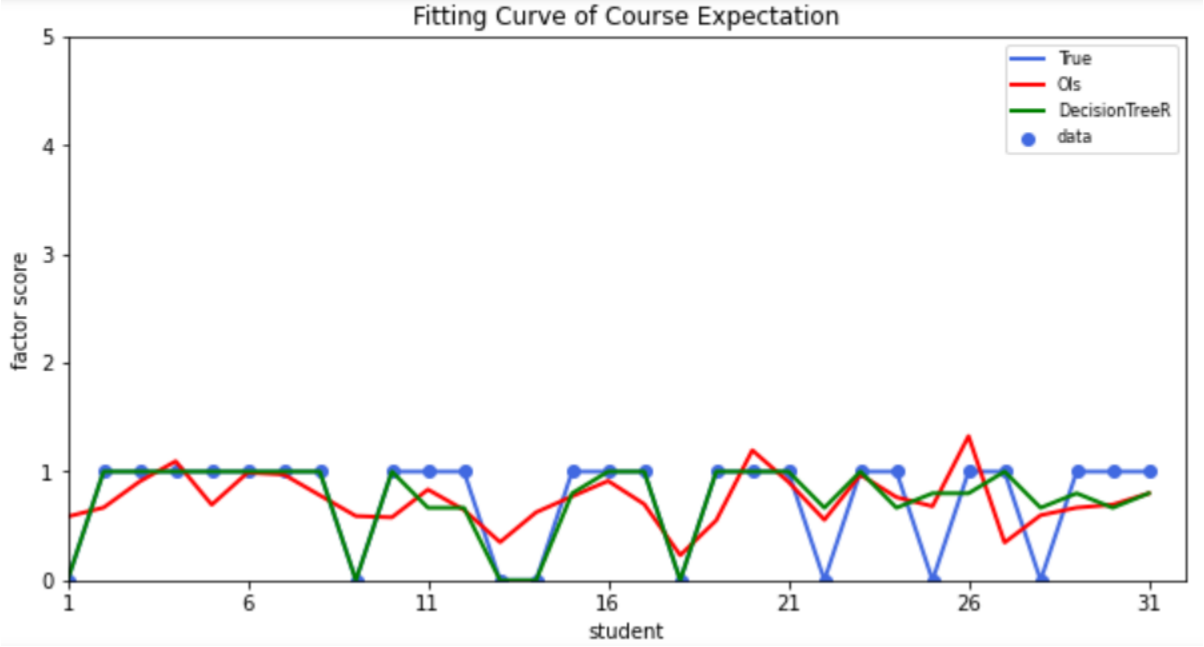
离人工智能只差一个 scikit-learn的距离

pip install scikit-learn

课程期望预测:机器学习 VS 回归

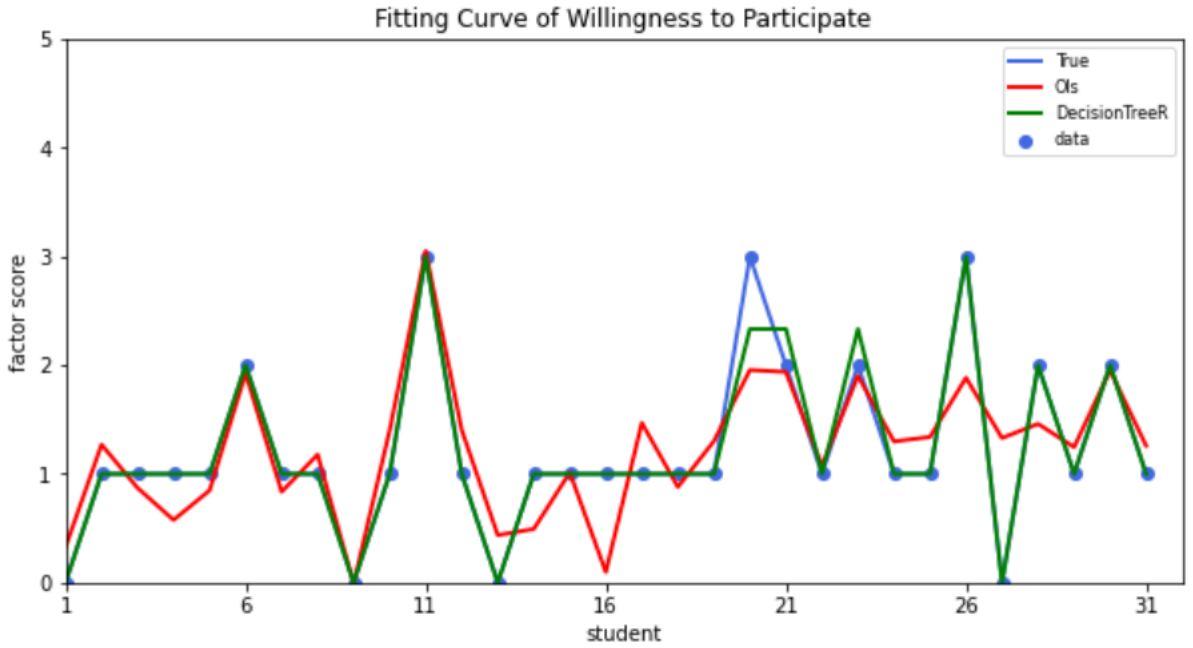
	OLS	DT
rmse	0. 37	0. 26

RMSE= $\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$



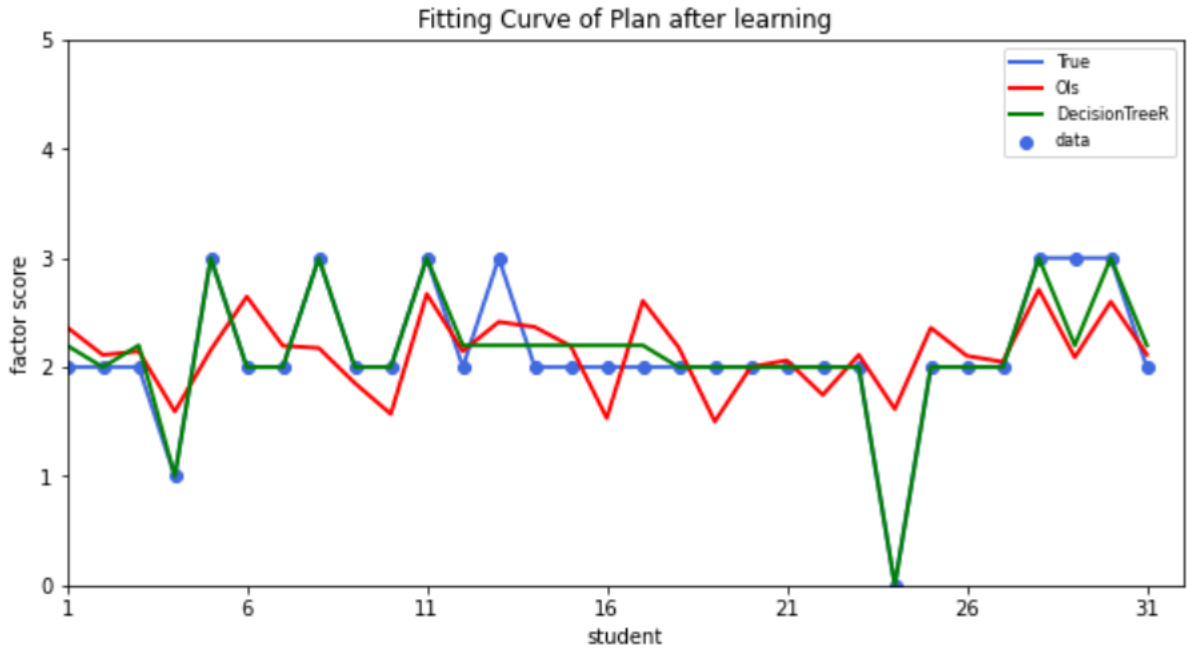
课程参与度预测:机器学习 VS 回归

	OLS	DT
rmse	0.48	0.15

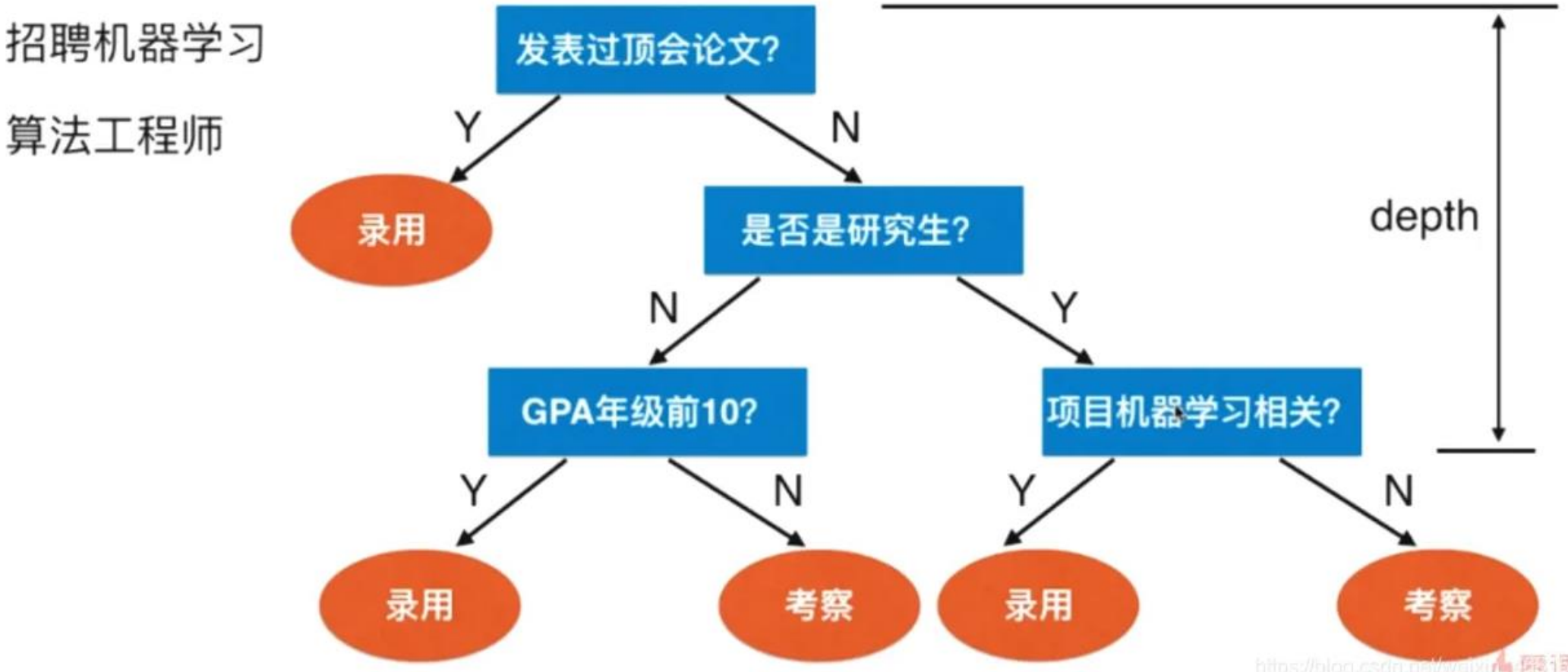


课后计划（从事相关学习或工作）预测:
机器学习 VS 回归

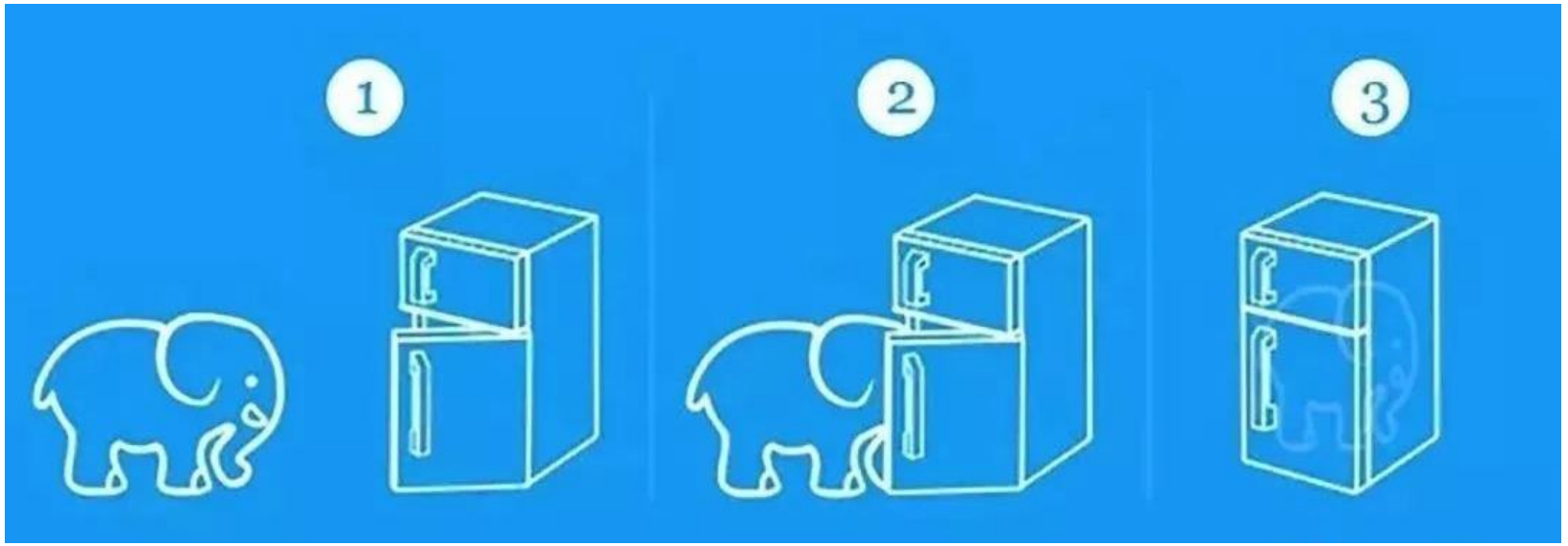
	OLS	DT
rmse	0.51	0.23



管导的决策树，财务的资金优化，人工智能的决策树和神经网络



把大象放进冰箱需要几步？



打开冰箱门：打开文件，读入数据

把大象放进冰箱：将数据和算法参数对应

关上冰箱门：算法跑结果