

统计学（Python实现）

第一次上机课

赵震宇（2120253538）

南开大学 国际经济研究所

zzynankai@outlook.com

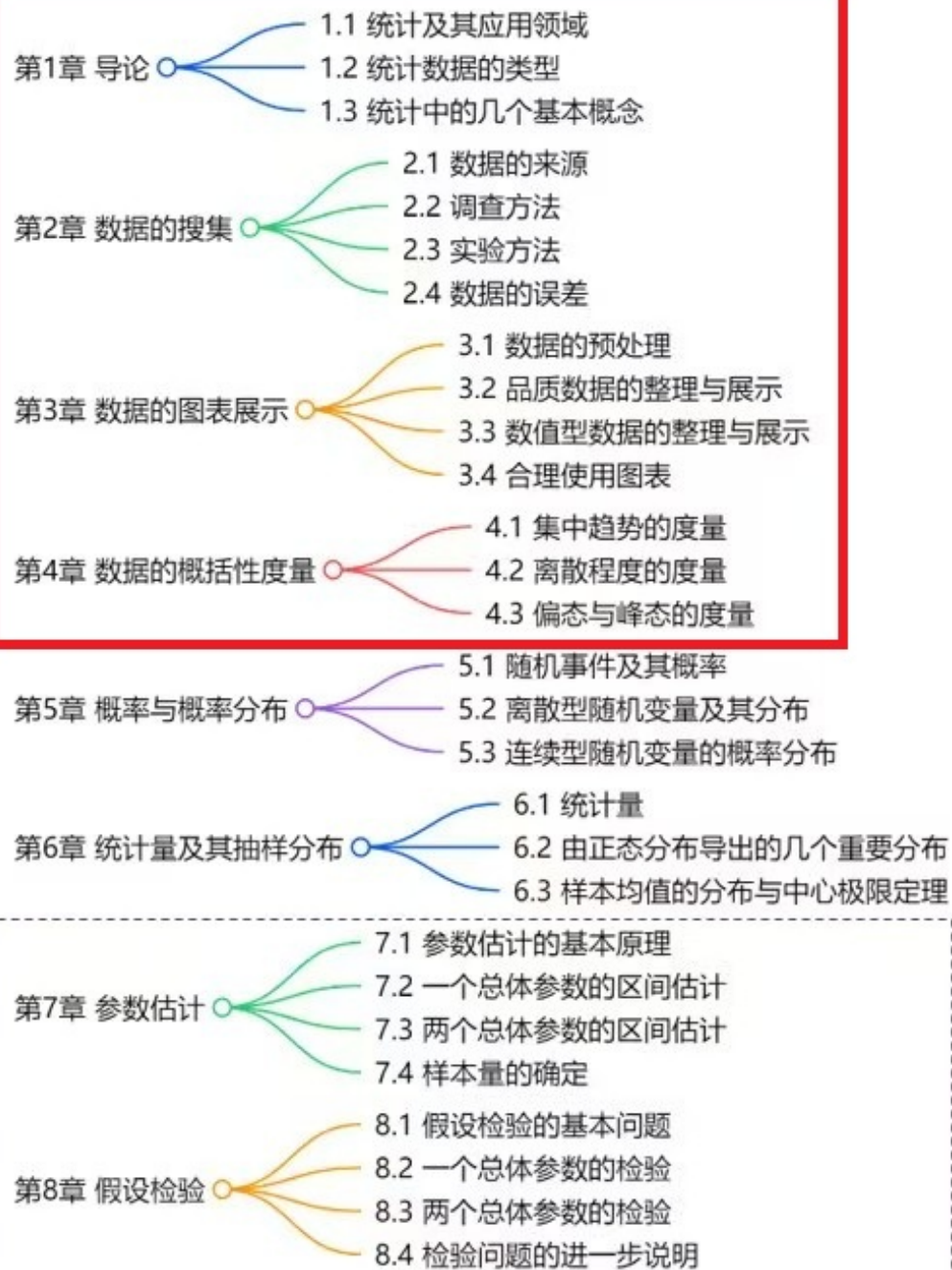
xishanyu2.github.io

2025 年 10 月 9 日

参考资料

- 贾俊平, 吴翌琳著. 统计学: 基于Python. 中国人民大学出版社. 2024
- 贾俊平编著. 统计学: Python实现. 高等教育出版社. 2021
- Python for Data Analysis, 3E ↗
- 鸢尾花书 ↗





CONTENT

目录

1 Python基础（不讲） ↗

2 数据可视化 ↗

3 描述性统计 ↗

4 课后习题 ↗

1. Python基础（不讲）

1.0 运行环境

- Anaconda Navigator → JupyterLab/Notebook
- Cursor – The AI Code Editor ↗

1.1 遇到问题怎么办？

- 问AI
- Stack Overflow ↗

1.2 数据读取和保存

```
import pandas as pd
table1 = pd.read_csv("F:/pydata/example/chap01/table1.csv")
table1 #默认为“UTF-8”格式
```

```
pip install openpyxl
import pandas as pd
df = pd.read_excel("F:/pydata/example/chap01/table3.xlsx")
df
```

- 保存为新数据时，需要指定文件类型：.csv/.xlsx
- 保存为csv类型时还需指定编码格式：utf-8/gbk

```
table1.to_csv("F:/pydata/example/chap01/df1.csv",index=False, encoding='gbk')
```

1.3 6种基本数据结构

1. 数字 (number)
2. 字符串 (string)
3. 元组 (tuple)
4. 列表 (list)
5. 字典 (dictionary)
6. 集合 (set)

“ 数字类型：整数型 (int)、浮点型 (float)、布尔型 (bool)、复数型 (complex)

指定字符串：' '、" "、''' '''

字典内含键 (key) 值 (value) 对

名称	列表	元组	集合	字典
表示方式	[a, b, c]	(a, b, c)	{a, b, c}	{a:x, b:y}
空	[]	()	set()	{ }或 dict()
其他创建方式	ls=list((1,'one')), 外围()表示list的参数列表, 内层()为元组	t=tuple([1,'one']), 外围参列, 内层列表。单个元素的元组需要在元素后添加,		用 dict 函数, ()中加入元素, 用, 隔开
访问元素	通过下标, 访问单个获得元素, 访问多个获得列表	通过下标, 访问单个获得元素, 访问多个获得列表		通过键进行访问
				dMerge=dict(d1,**d2)
拼接/重复	+/*	+/*		
元素可否重复	可	可	不可	可
修改	通过下标方式通过变量修改列表会导致另一个变量发生改变, 可通过列表截取[:]的方式修改单一变量	不可修改		见插入
更改指向对象	通过使用 copy 模块提供的 deepcopy 函数复制变量, 可以使两个变量指向不同的对象			浅拷贝 d.copy() 深拷贝 copy, deepcopy(d)

查找	<code>ls.index(元素值)</code>			<code>d.get('键')</code> 判断键的存在性
插入	<code>ls.insert(指定位置, x)</code> , <code>ls.append(x)</code> 则是将 <code>x</code> 插在列表末端, <code>x</code> 可以是元素也可以是列表		<code>s.add(x)</code> <code>x</code> 可哈希或 <code>s.update(x)</code> <code>x</code> 可迭代	字典. <code>update(字典)</code> 若插入的键名不存在, 则可直接字典['键名']='值'
删除	使用 <code>del</code> 语句, 可以类似访问元素的方式进行删除			<code>del d['键']</code> 或 <code>d.pop('键', 'x')</code> , 若不存在名为键的元素, 则返回 <code>x</code> 的值
最大/最小元素	<code>max/min(ls)</code>	<code>max/min(t)</code>	交集 <code>s1.intersection(s2)</code> 并集 <code>s1.union(s2)</code> 差集 <code>s1.difference(s2)</code> 对称差集 <code>s1.symmetric_difference(s2)</code> 子集 <code>s1.issubset(s2)</code> 父集 <code>s1.issuperset(s2)</code>	<code>d.clear()</code> 可以清除字典中所有元素
统计元素个数	<code>ls.count(统计值)</code>			<code>len(d)</code>
计算长度	<code>len(ls)</code>			获取键的集合 <code>d.keys()</code>
元素排序	<code>ls.sort(key=None, reverse=False)</code> 升序 F (默认), 降序 T, None 可用 <code>lambda</code> 函数的定义式(<code>lambda</code> 形参: 对象属性), 也可以直接定义新函数			获取值的集合 <code>d.values()</code>
				遍历字典 <code>d.items()</code>

创建列表

```
ls1 = ['甲', 23, True, [1,2,3]]  
ls2 = list(range(10)) #起始为0  
ls3 = list(range(1,20,2)) #左闭右开
```

列表操作

```
l1 = [1,2,4]  
l1.append(3) #加入元素到列表尾部  
l1.sort() #排序  
l1.insert(2,5) #在第2个位置插入5 (python从第0位开始)  
l1.pop(3) #移除第三个位置的元素并返回其值
```

```
l2 = ['甲', '乙', '丙', '丁']  
l12 = l1 + l2
```

字典操作

```
dc1 = {'甲':68, '乙':85, '丙':74, '丁':88}
dc2 = {'甲'=68, '乙'=85, '丙'=74, '丁'=88}
dc1.keys()
dc1.values()
dc1.items()
dc1['甲']
del dc1['甲']
```

集合操作

```
set1 = set{2,2,2,1,8,3,3,5,5}
set2 = set{2,2,2,1,4,3,3,5,6,6}
set1&set2 #交集, 或写成set1.intersection(set2)
set1|set2 #并集, 或写成set1.union(set2)
```

1.4 Python“三驾马车”：数学运算和可视化库

- NumPy [↗](#)
- pandas – Python Data Analysis Library [↗](#)
- Matplotlib — Visualization with Python [↗](#)

1.4.1 numpy 中的数组 (array)

```
# 一维数组—向量 (vector)
import numpy as np
a1 = np.array([5,4,1,2,3])
a2 = np.arange(12)
a3 = np.arange(2,6,0.5)

# 二维数组—矩阵 (matrix)
a4 = np.array([[1,2],[3,4],[5,6]])

a5 = a2.reshape(3,4) #改向量为矩阵
a5[2] #元素为向量
```

1.4.2 pandas 中的序列 (series)

创建序列

```
import pandas as pd
s1 = pd.Series([1,2,3,4]) #默认索引为0 1 2 3
s2 = pd.Series([5,6,7,8], index=['a','b','c','d'])
s2[1]
s2[[1,2]]
s2[['b','c']]
s3 = pd.Series(range()5)
```

序列操作

```
print('索引: ',s2.index)
print('数据: ',s2.values)
print('类型: ',s2.dtype)
```

```
s2.name = '序列6'  
s2.index.name = '索引名'  
s2 = s6.astype(float)  
s2[[1,3]] = [2,8]
```

```
# 序列计算  
s3 = pd.Series([1,2,3],index=['a','c','e'],dtype=float)  
s2 + s3 #有共同索引的才能相加，否则为NaN  
c = s2.cumsum() #累计求和  
s = s2.sum()  
m = s2.mean()  
print("累加: ", '\n', c, '\n', "总加=", s, '\n', '平均数=', m)
```

1.4.3 pandas 中的数据框（data frame）——类似Excel数据表

```
# 创建数据框
import pandas as pd
d = {"姓名":["甲","乙","丙","丁","戊"],
     "统计学":[68,85,74,88,63],
     "数学":[85,91,74,100,82],
     "经济学":[84,63,61,49,89]}
table1_1 = pd.DataFrame(d)
table1_1
table1_1.head()
table1_1.tail(3)
table1_1.T #转置
```

数据框操作

```
import pandas as pd
df = pd.read_csv("F:/pydata/example/chap02/table2_1.csv")
df[['数学']] #为DataFrame, 而df['数学']为Series, 后者等价于df.数学
df.loc[2]
df.loc[[2,4]]
df.loc[2:4]
dd = df.iloc[[1,3],[0,2,3]]
df[:]
df[2:]
df[:,2] #隔行取值: 0 2 4

df["会计学"]=[88,75,92,67,78]
df.insert(2,'会计学',[88,75,92,67,78])
df.drop(labels='数学',axis=1,inplace=True) #axis=1表示列, axis=0表示行
#inplace=True参数用于直接修改原始对象, 而不是创建一个新的对象。
df.drop(index=2,inplace=True)

df.rename(columns={'数学':'计算机','经济学':'管理学'})
df.iloc[2,1]=85
```

数据框排序

```
import pandas as pd
```

```
table2_1 = pd.read_csv("F:/pydata/example/chap02/table2_1.csv")
```

```
table2_1.sort_values(by='统计学',ascending=False) #False为降序
```

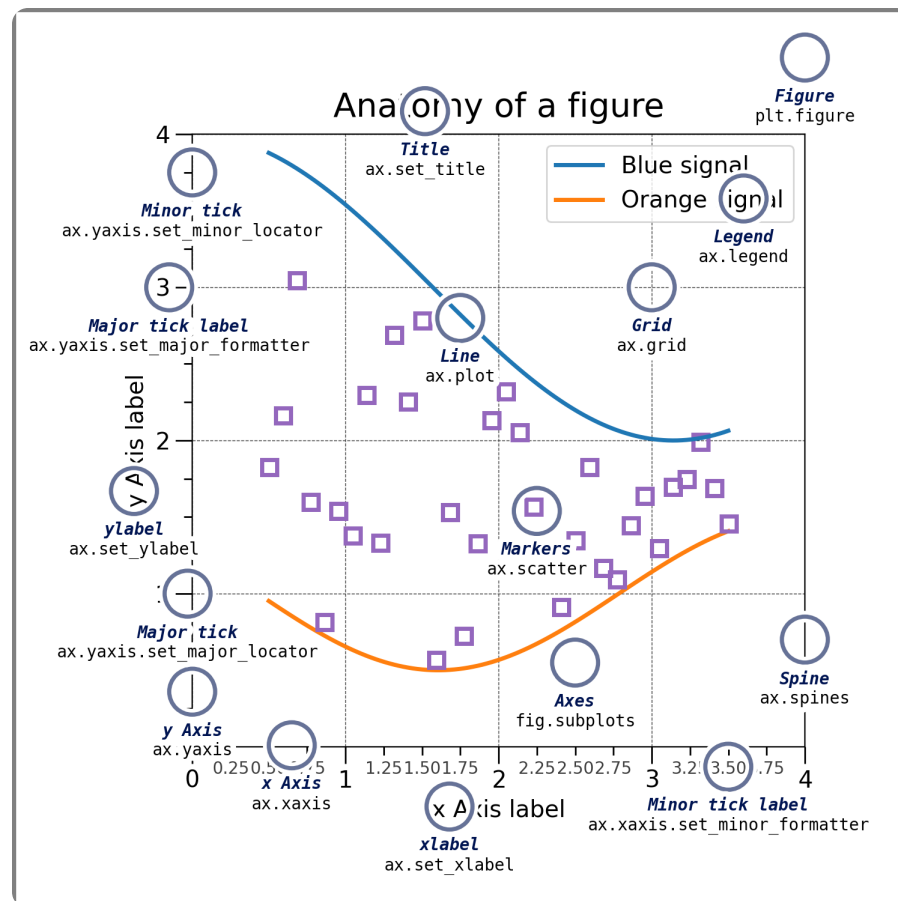
数据框合并

```
table_merge = pd.concat([table1,table2]).reset_index(drop=True)
```

2. 数据可视化

2.0 matplotlib 绘图

- 两个对象：figure（画布），axes（画像）

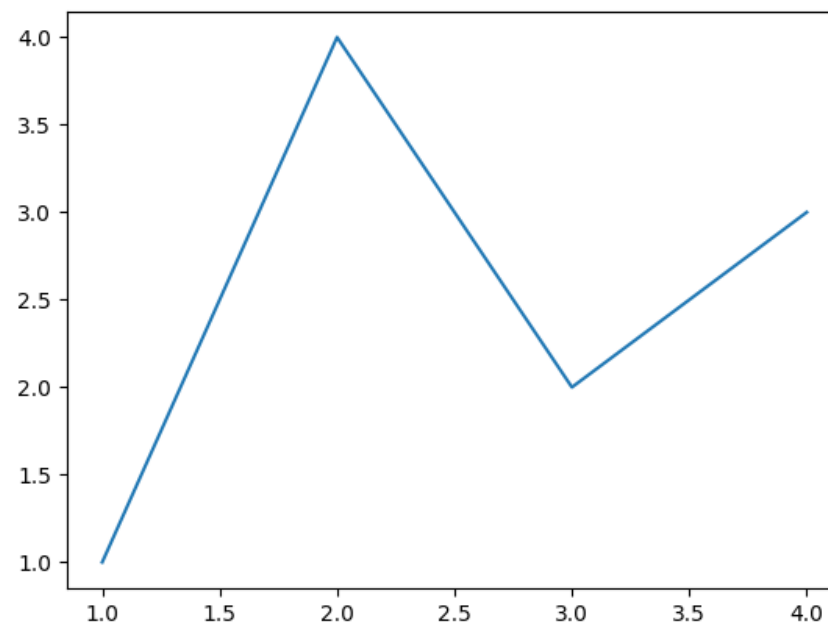


2.1 matplotlib入门

```
# matplotlib通常的引入约定是:  
import matplotlib.pyplot as plt
```

- Matplotlib在Figure上绘制数据，每个Figure可以包含一个或多个Axes，Axes是一个可以使用x-y坐标（或极坐标中的theta-r，3D图中的x-y-z等）指定点的区域。
- 创建一个包含单个Axes的Figure的最简单方法是使用plt.subplots。
- 然后可以使用ax.plot在Axes上绘制一些数据，并使用plot.show显示图形。

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.show()
```



暂停一下：先搞明白plt./ax./fig再画图

- 同样一个目标，在matplotlib中有N种不同的方法实现。
- Matplotlib为作图提供了两套可视化接口，分别对应MATLAB的陈述式语法和面向对象写法：
 - i. `plt.plot()`系列：适用于快速出图
 - ii. `fig, ax = plt.subplots(); ax.plot()`系列：适用于精细绘图
- 在Matplotlib官网搜索，通常能看到两套接口：
 - 结果中的`axes.Axes.pie`对应`ax.pie()`，`pyplot.pie`对应`plt.pie()`。

Search Results

Search finished, found 59 page(s) matching the search query.

-  **matplotlib.axes.Axes.pie** (Python method, in matplotlib.axes.Axes.pie)
-  **matplotlib.pyplot.pie** (Python function, in matplotlib.pyplot.pie)

🐛 爬虫导航

2.1.1 Figure和Subplot

matplotlib的图像都位于Figure对象中，可以用plt.figure创建一个新的Figure：

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

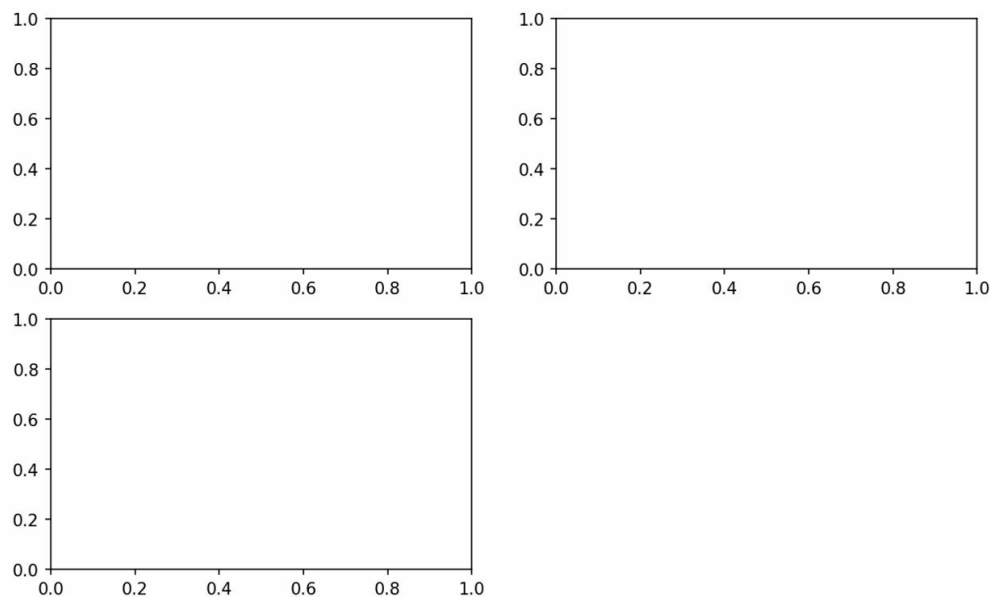
Out: <Figure size 640x480 with 0 Axes>

不能通过空Figure绘图，必须用add_subplot创建一个或多个subplot才行：

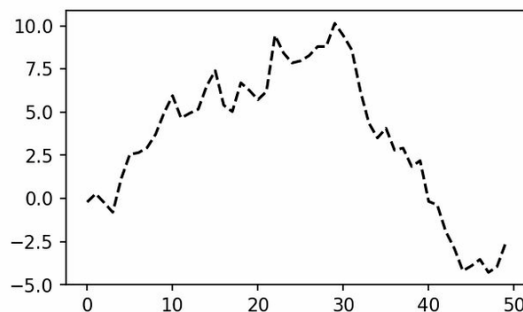
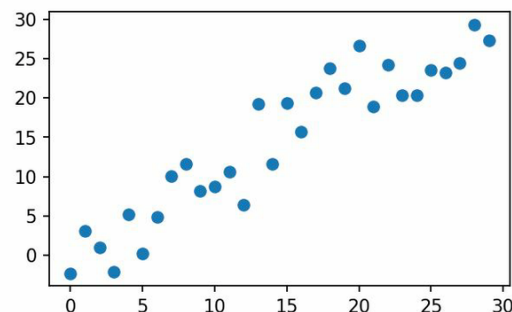
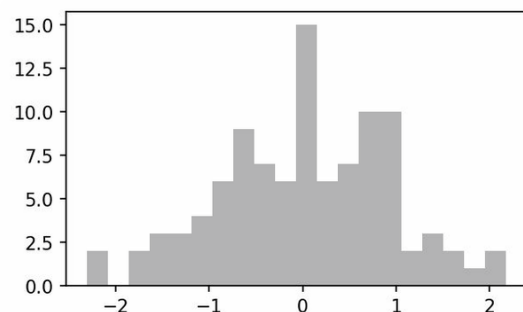
```
ax1 = fig.add_subplot(2, 2, 1)
```

subplot中参数的意思是：图像应该是 2×2 的（即最多4张图），且当前选中的是4个subplot中的第一个（编号从1开始）。如果再把后面两个subplot也创建出来，最终得到的图像如下：

```
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```



```
ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
ax3.plot(np.random.randn(50).cumsum(), 'k--')
```



matplotlib中更为方便的方法是`plt.subplots`，可以创建一个新的Figure，并返回含有已创建的subplot的NumPy数组；就可以轻松地对axes数组进行索引，例如`axes[0,1]`

```
fig, axes = plt.subplots(2, 3)
axes
```

参数	说明
nrows	subplot的行数
ncols	subplot的列数
sharex	所有subplot应该使用相同的X轴刻度（调节xlim将会影响所有subplot）
sharey	所有subplot应该使用相同的Y轴刻度（调节ylim将会影响所有subplot）
subplot_kw	用于创建各subplot的关键字字典
**fig_kw	创建figure时的其他关键字，如 <code>plt.subplots(2,2,figsize=(8,6))</code>

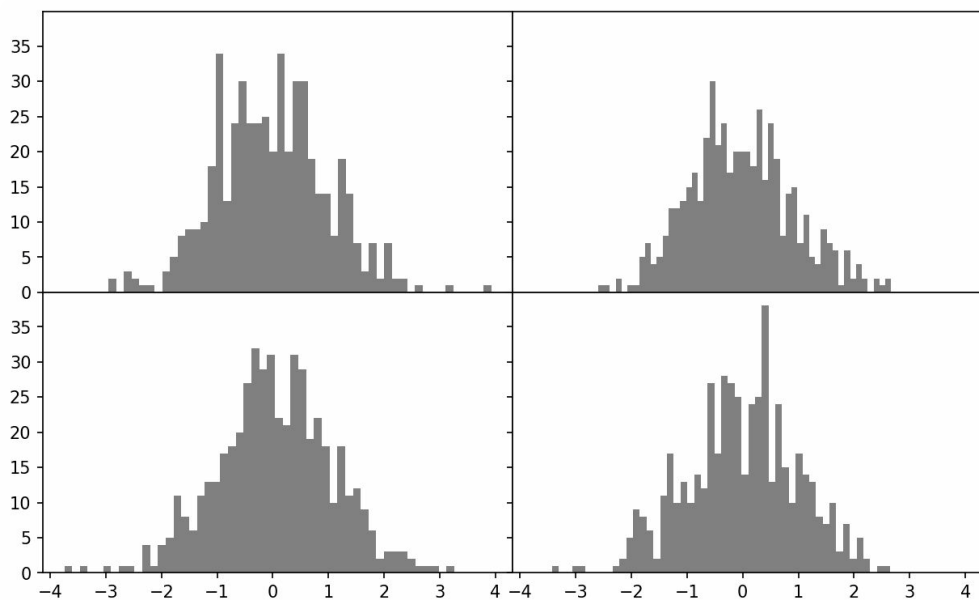
2.1.2 调整subplot周围的间距

默认情况下，matplotlib会在subplot外围留下一定的边距，并在subplot之间留下一定的间距。

利用Figure的subplots_adjust方法可以修改间距，其中wspace和hspace用于控制宽度和高度的百分比，可以用作subplot之间的间距：

```
subplots_adjust(left=None, bottom=None, right=None, top=None,  
                wspace=None, hspace=None)
```

```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5) # alpha指定直方图的透明度
plt.subplots_adjust(wspace=0, hspace=0)
```



2.1.3 颜色、线型和标记

我们之前看到过k--, 也可以拆分为color='k', linestyle='--'

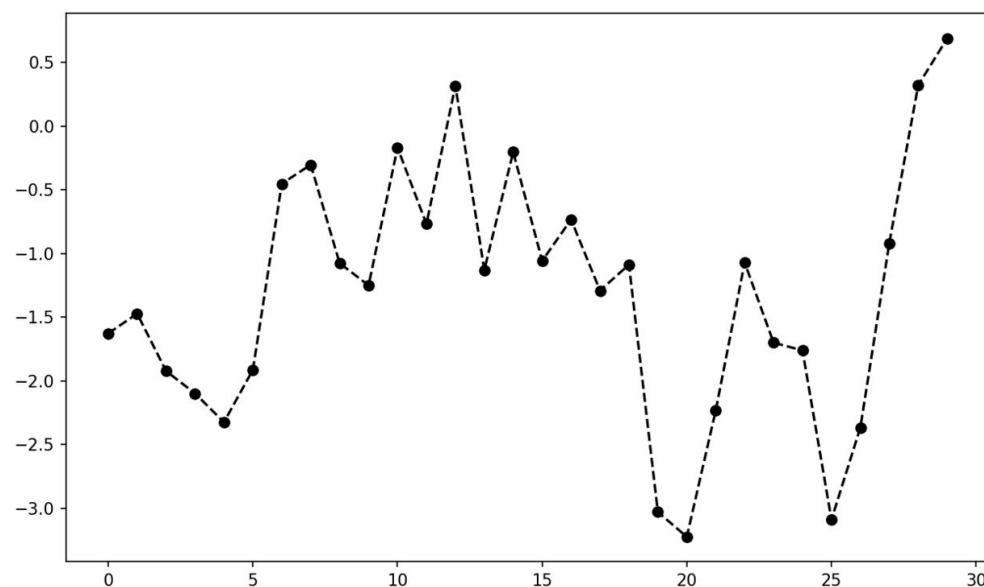
线的颜色可以使用color参数来定义, 简写为c:

颜色	'r'	'g'	'b'	'c'	'm'	'y'	'k'	'w'
说明	红色	绿色	蓝色	青色	品红	黄色	黑色	白色

线的类型可以使用linestyle参数来定义, 简写为ls:

线型	'solid' (默认)	'dotted'	'dashed'	'dashdot'	'None'
简写	'-'	':'	'--'	'-.'	'' 或 ''
说明	实线	点虚线	破折线	点划线	不画线

```
from numpy.random import randn  
plt.plot(randn(30).cumsum(), 'ko--')
```



标记也可以放到格式字符串中，但标记类型和线型必须放在颜色后面。

```
plt.plot(randn(30).cumsum(), color='k', linestyle='dashed', marker='o')
```

marker部分可以定义的符号如下：

标记	"."	"o"	"v"	"^"	"<"	">"	"s"	"d"	"D"	"*"	"None", " " or ""
说明	点	实心圆	下三角	上三角	左三角	右三角	正方形	瘦菱形	菱形	星号	没有任何标记

Source: Matplotlib 教程 | 菜鸟教程 [↗](#)

2.1.4 刻度、标签、标题

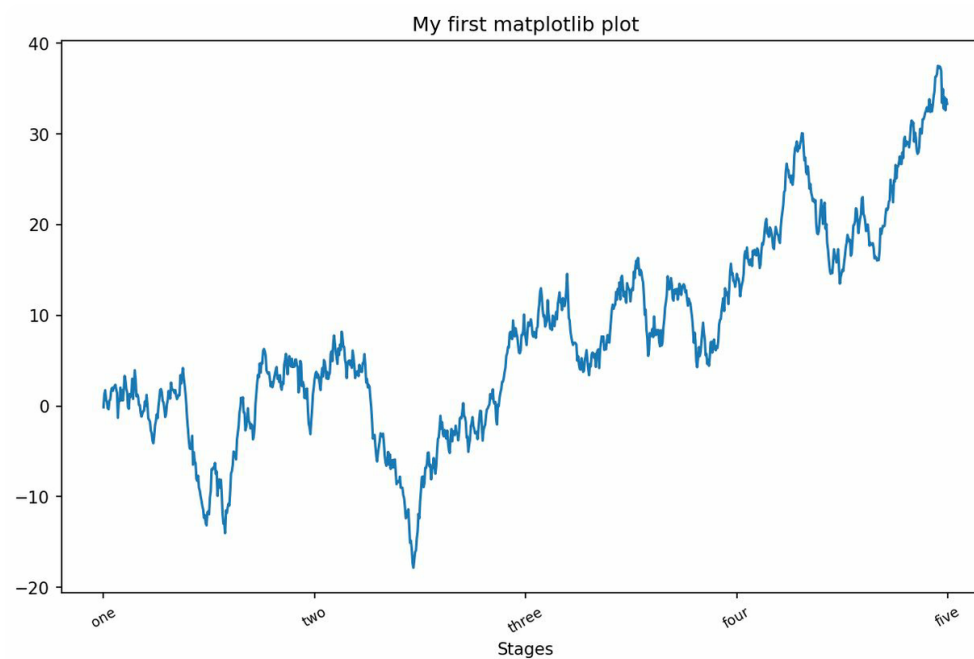
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(np.random.randn(1000).cumsum())
```

使用`set_xticks`和`set_xticklabels`改变x轴刻度，前者告诉matplotlib要将刻度放在数据范围中的哪些位置，默认情况下，这些位置也就是刻度标签；通过`set_xticklabels`将任何其他值用作标签：

```
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                             rotation=30, fontsize='small') # rotation选项设定x刻度标签倾斜30度
```

用`set_xlabel`为X轴（Y轴类似）设置一个名称，并用`set_title`设置一个标题：

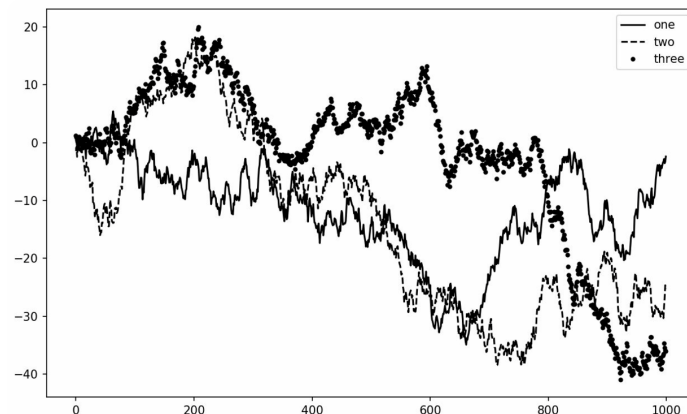
```
ax.set_title('My first matplotlib plot')  
ax.set_xlabel('Stages')
```



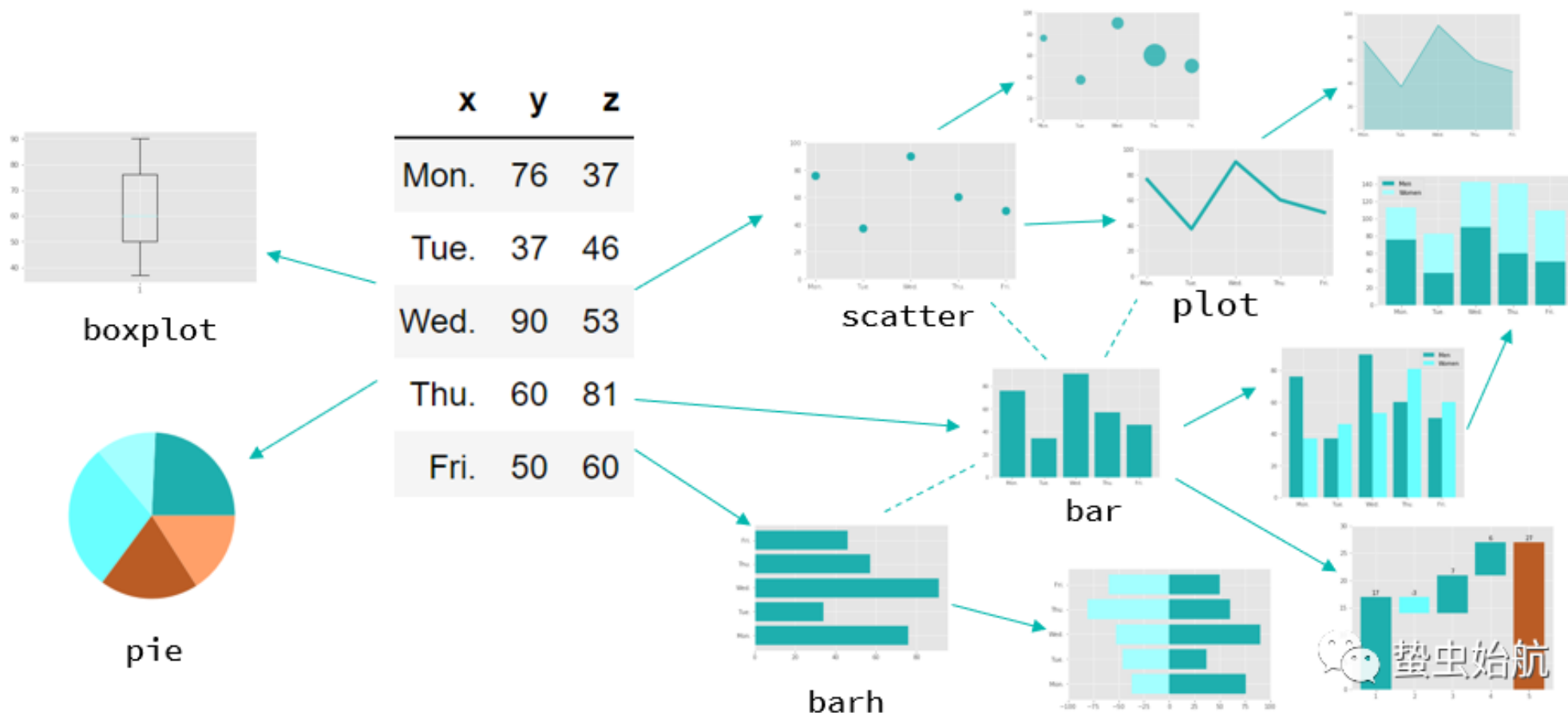
2.1.5 图例

调用`ax.legend()`或`plt.legend()`来自动创建图例，`loc`告诉`matplotlib`要将图例放在哪。

```
from numpy.random import randn
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'k', label='one')
ax.plot(randn(1000).cumsum(), 'k--', label='two')
ax.plot(randn(1000).cumsum(), 'k.', label='three')
ax.legend(loc='best')
```



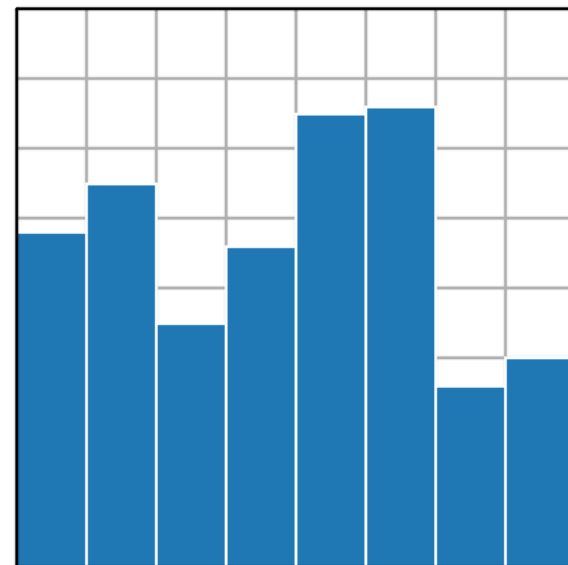
将数据映射为可视图表

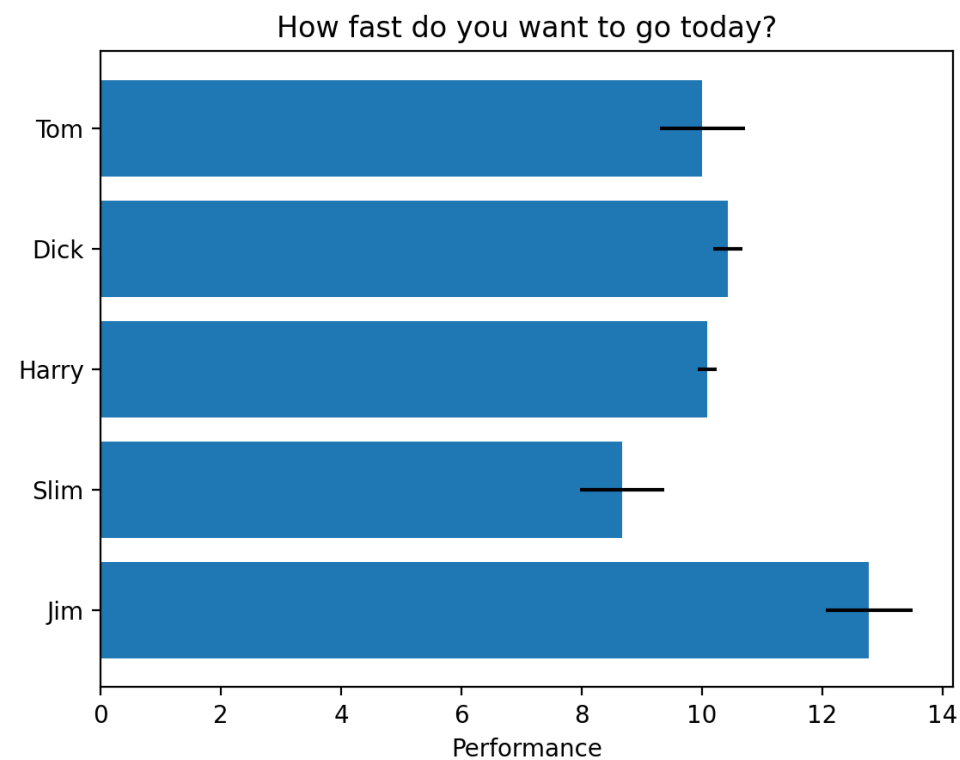
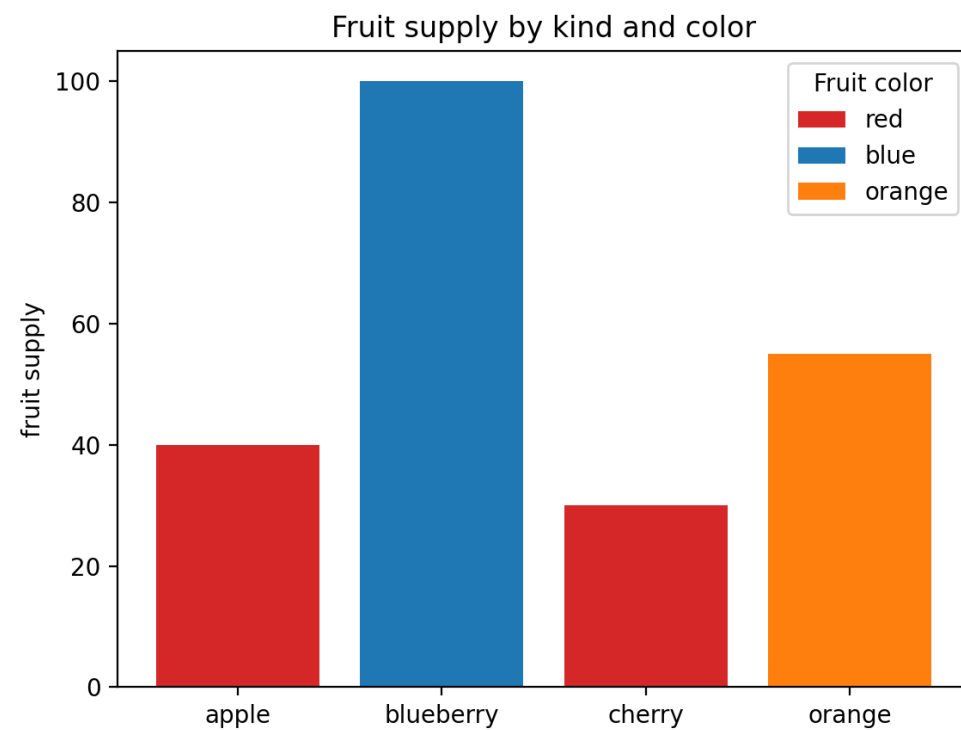


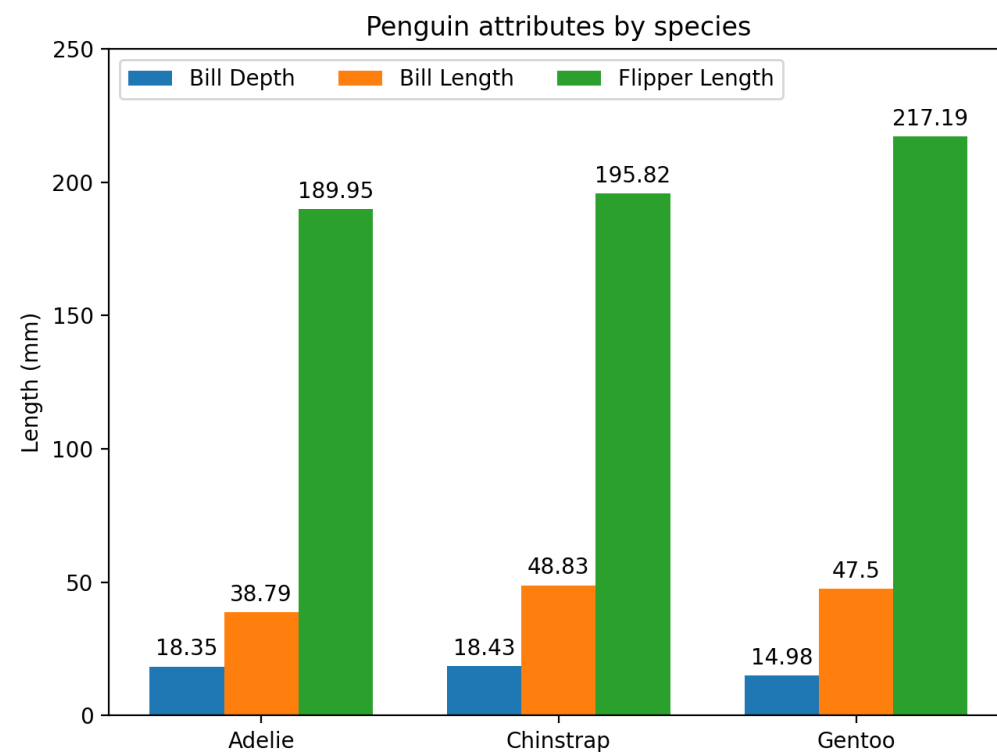
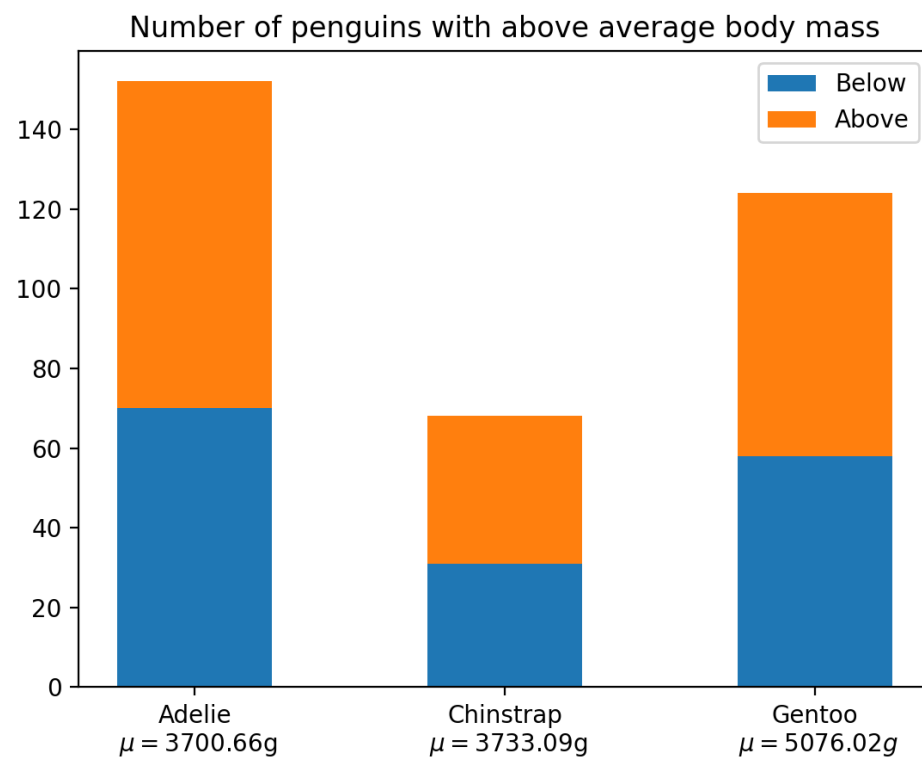
2.2 类别数据可视化

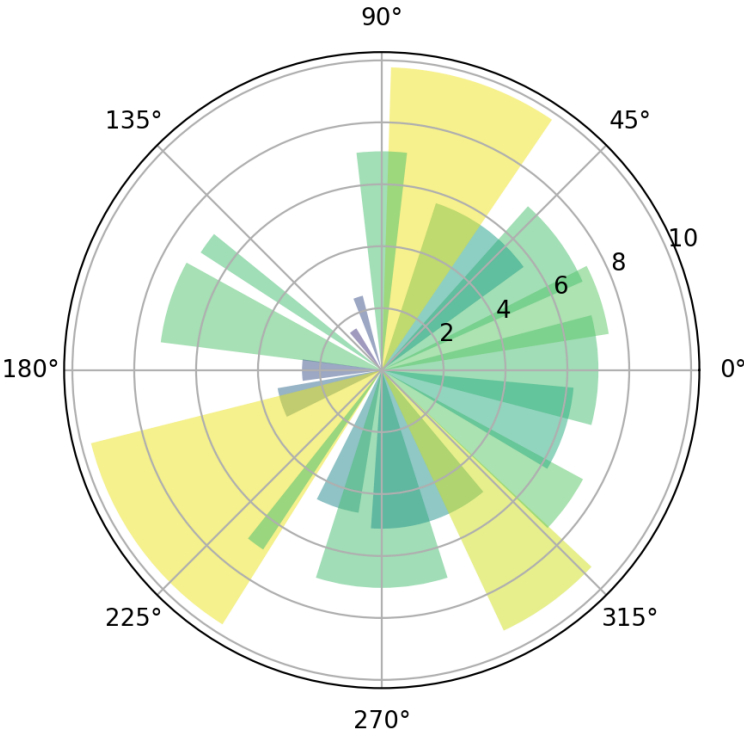
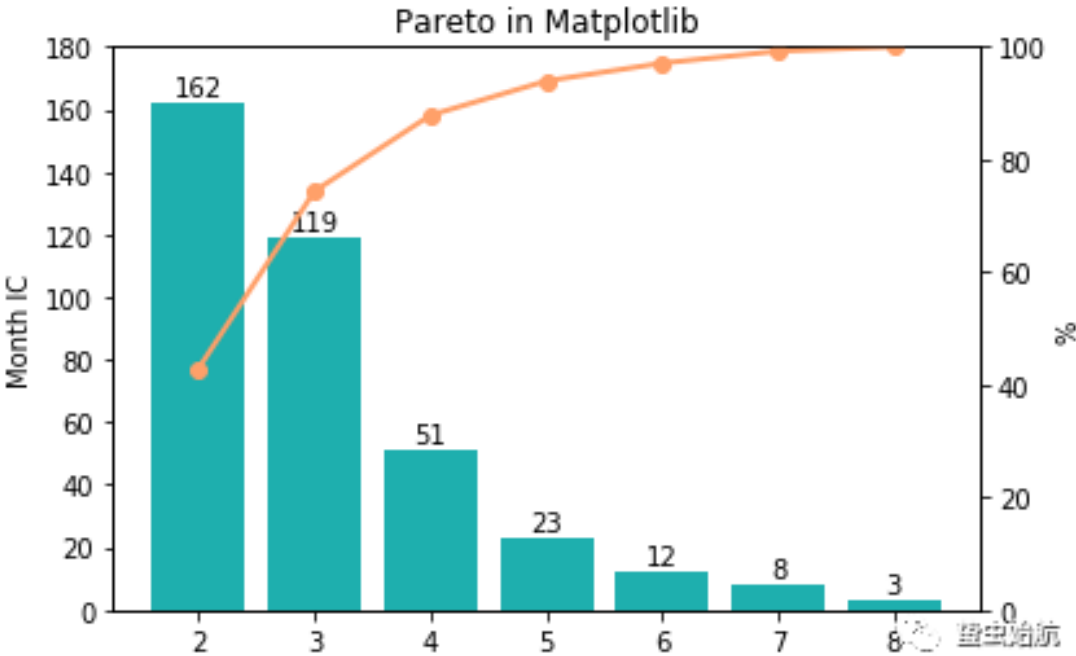
条形图

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
x = 0.5 + np.arange(8)
y = [4.8, 5.5, 3.5, 4.6, 6.5, 6.6, 2.6, 3.0]
fig, ax = plt.subplots()
ax.bar(x, y, width=1, edgecolor="white", linewidth=0.7)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
        ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```



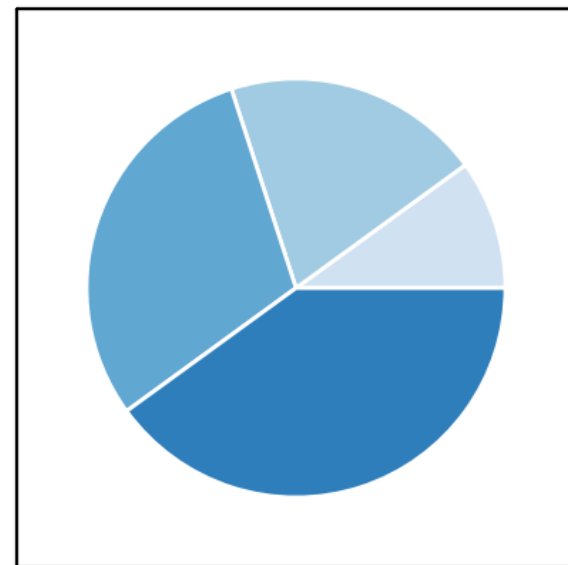


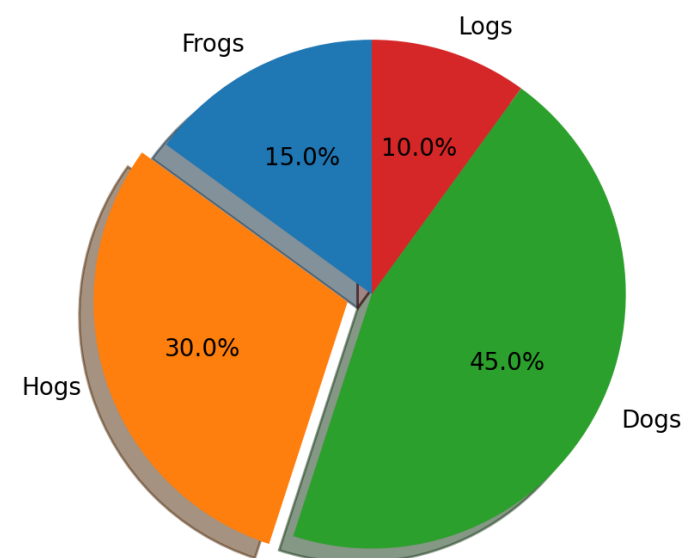
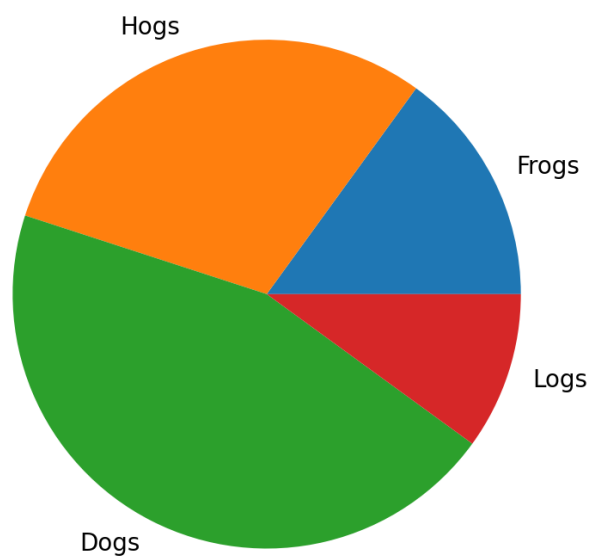


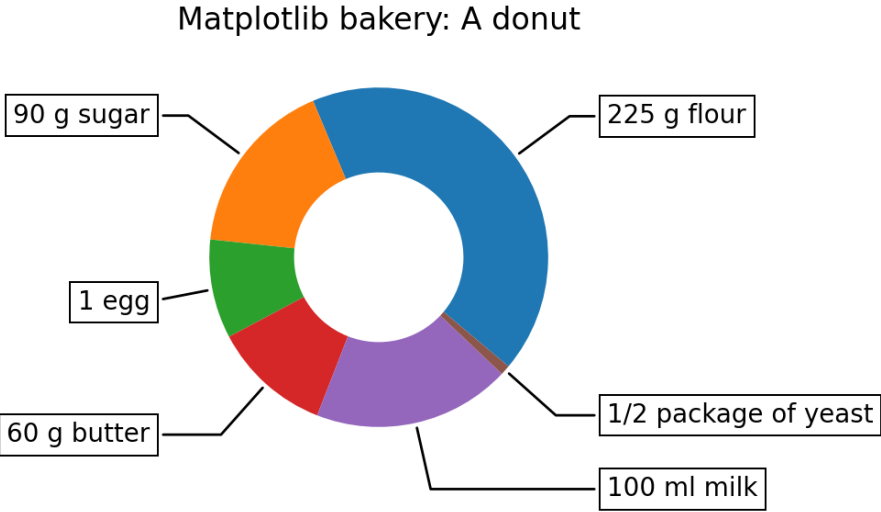
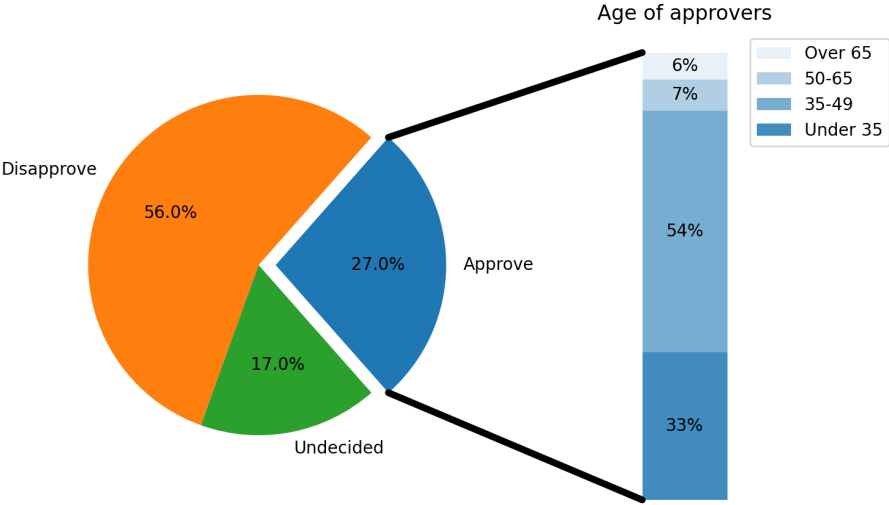


饼图

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery-nogrid')
x = [1, 2, 3, 4]
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(x)))
fig, ax = plt.subplots()
ax.pie(x, colors=colors, radius=3, center=(4, 4),
      wedgeprops={"linewidth": 1, "edgecolor": "white"},
      frame=True)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
      ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```



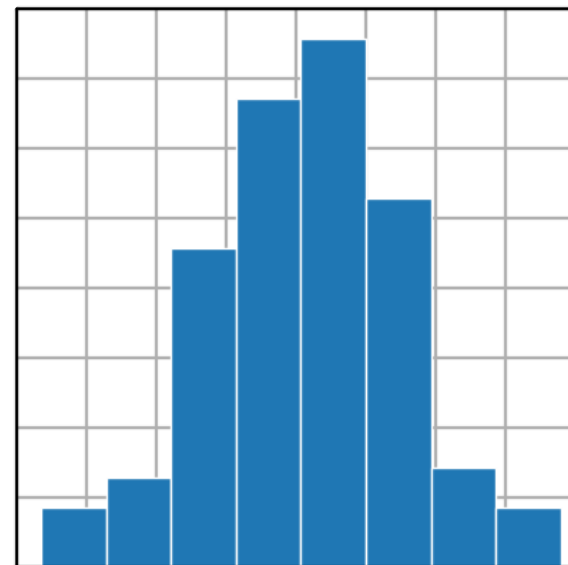


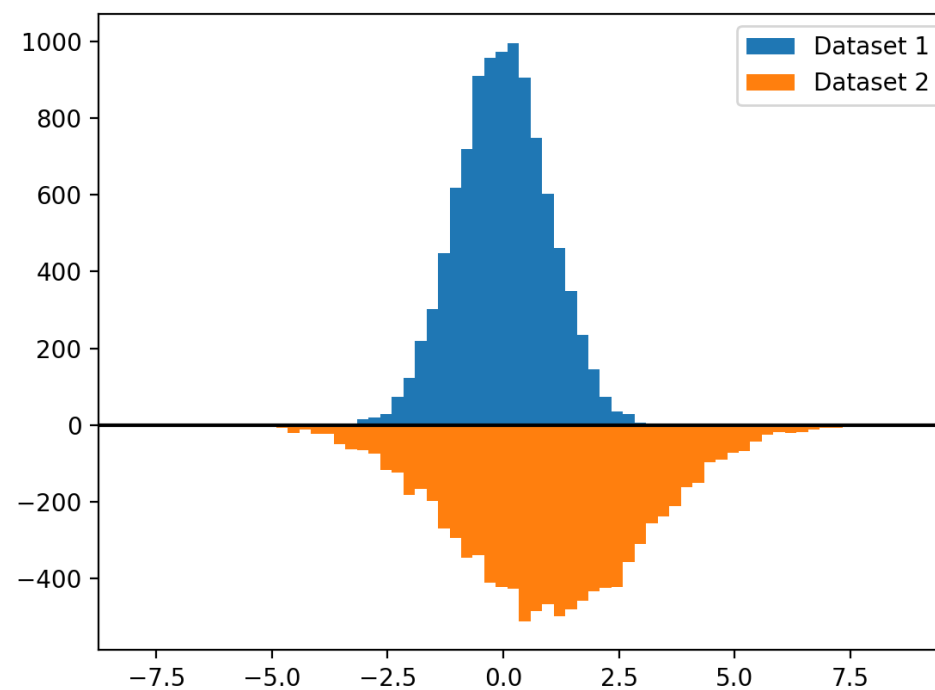
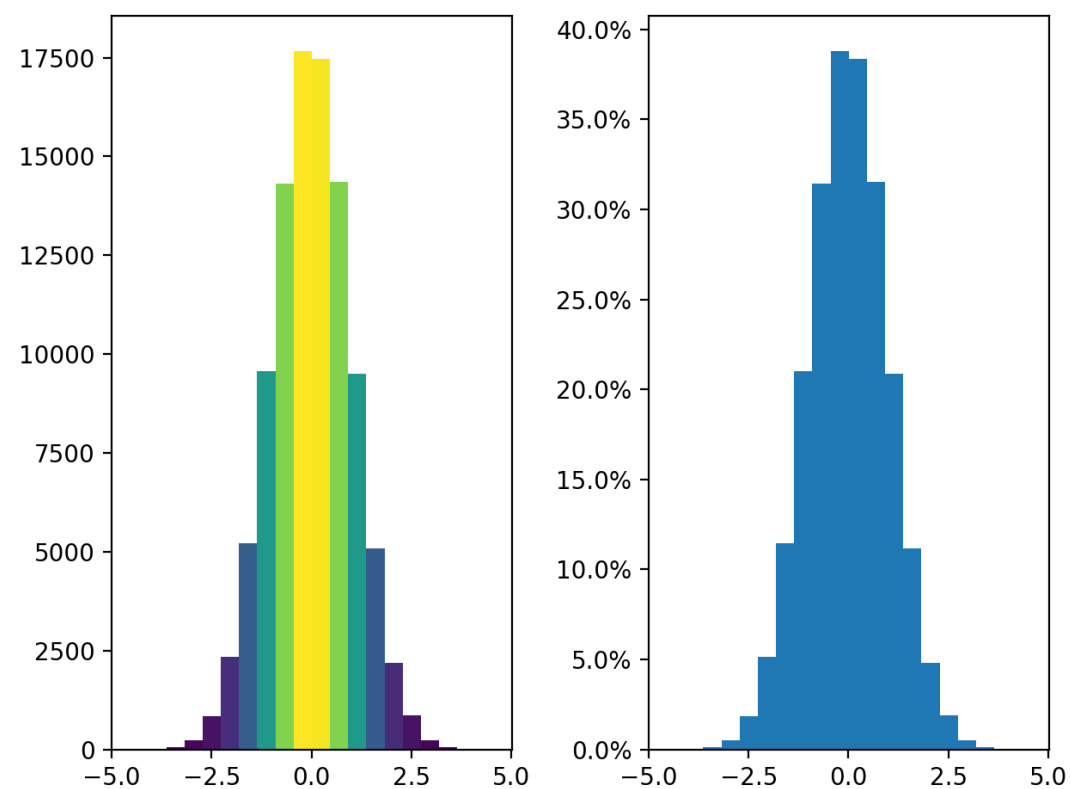


2.3 数据分布可视化

直方图

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
np.random.seed(1)
x = 4 + np.random.normal(0, 1.5, 200)
fig, ax = plt.subplots()
ax.hist(x, bins=8, linewidth=0.5, edgecolor="white")
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 56), yticks=np.linspace(0, 56, 9))
plt.show()
```



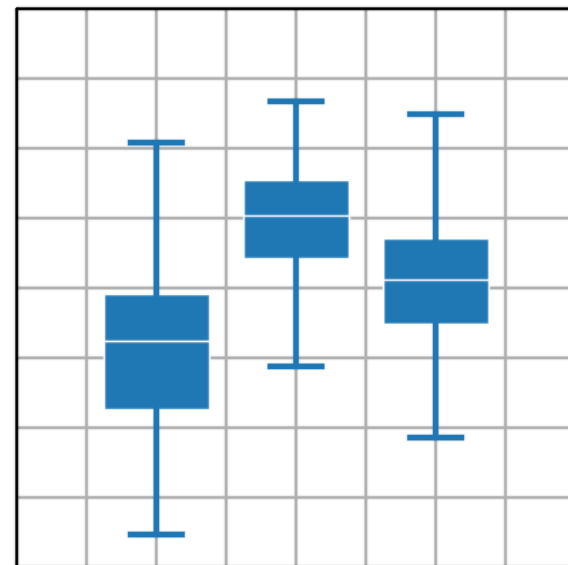


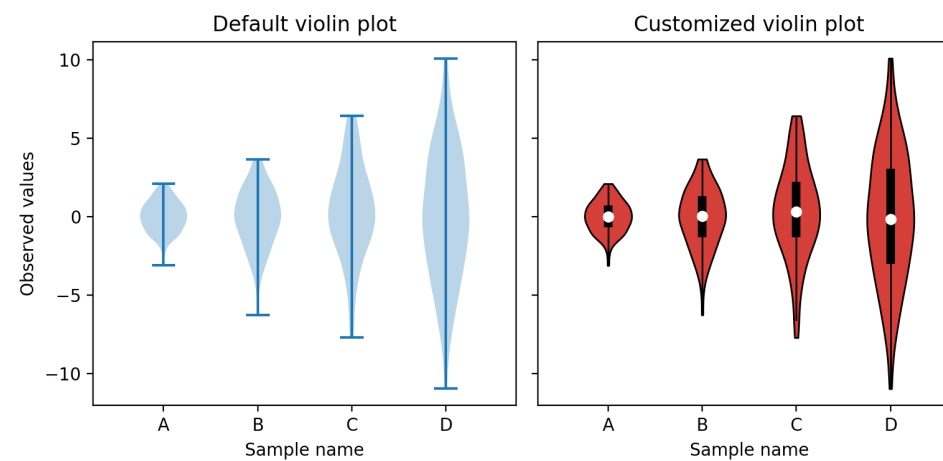
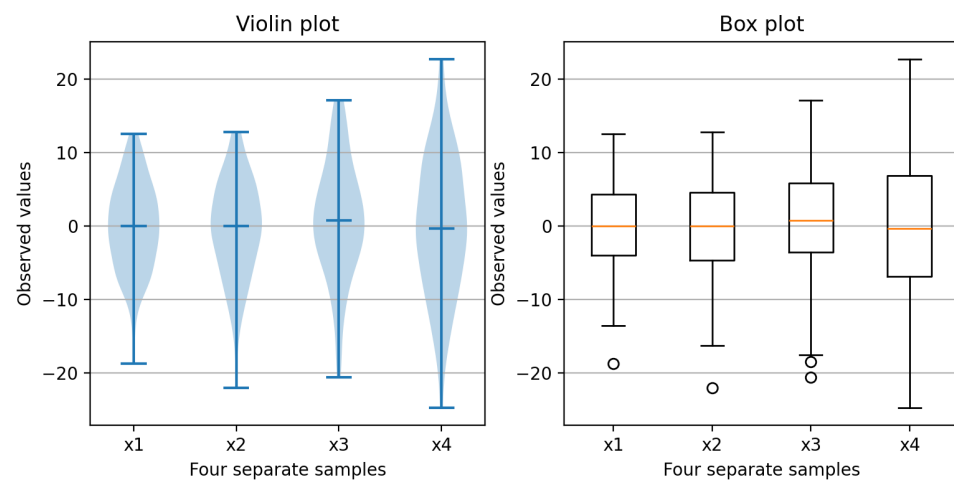
核密度图

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('_mpl-gallery')
np.random.seed(42)
data = pd.DataFrame({
    'Group1': np.random.normal(30, 5, 1000),
    'Group2': np.random.normal(40, 8, 1000),
    'Group3': np.random.normal(50, 10, 1000)
})
data.plot.kde()
plt.xlabel('value')
plt.legend(['Group1', 'Group2', 'Group3'], loc="best")
plt.title('KDE Plot')
plt.grid(True)
plt.show()
```

箱线图

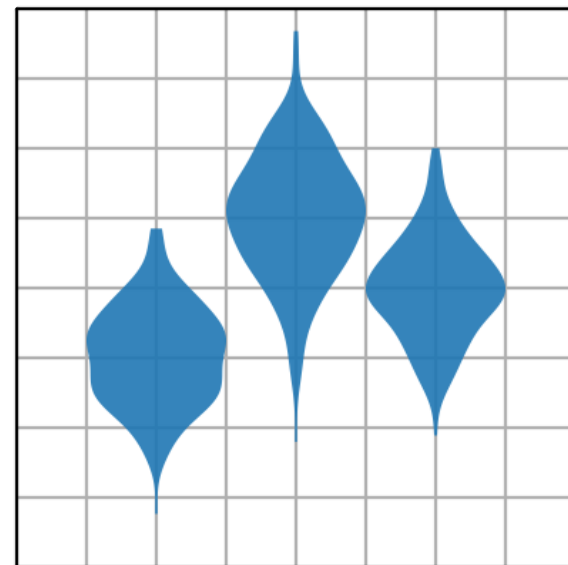
```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
np.random.seed(10)
D = np.random.normal((3, 5, 4), (1.25, 1.00, 1.25), (100, 3))
fig, ax = plt.subplots()
VP = ax.boxplot(D, positions=[2, 4, 6], widths=1.5,
    patch_artist=True, showmeans=False, showfliers=False,
    medianprops={"color": "white", "linewidth": 0.5},
    boxprops={"facecolor": "C0", "edgecolor": "white", "linewidth": 0.5},
    whiskerprops={"color": "C0", "linewidth": 1.5},
    capprops={"color": "C0", "linewidth": 1.5})
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
    ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```





小提琴图

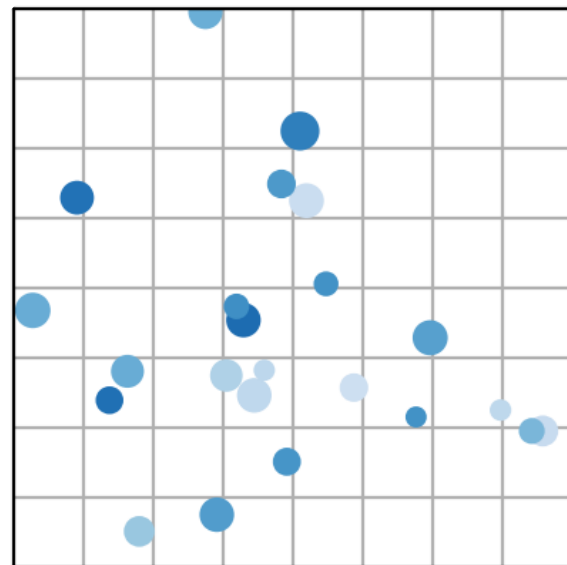
```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
np.random.seed(10)
D = np.random.normal((3, 5, 4), (0.75, 1.00, 0.75), (200, 3))
fig, ax = plt.subplots()
vp = ax.violinplot(D, [2, 4, 6], widths=2,
                    showmeans=False, showmedians=False, showextrema=False)
for body in vp['bodies']:
    body.set_alpha(0.9)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
        ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```



2.4 变量关系可视化

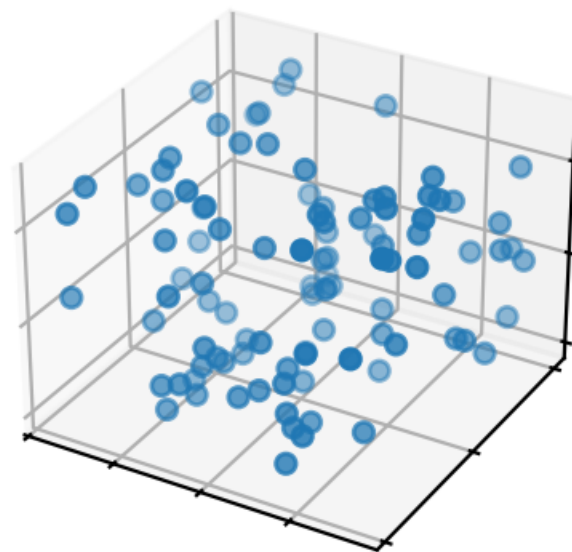
散点图

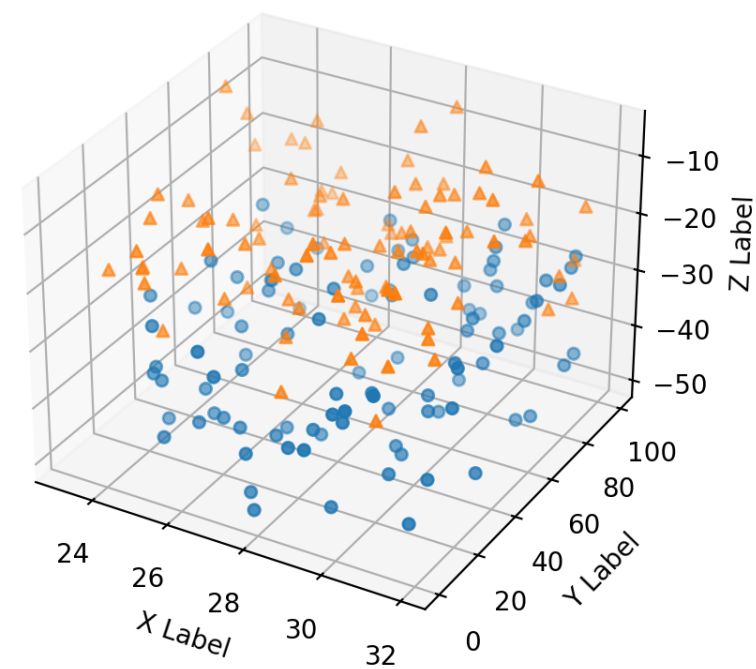
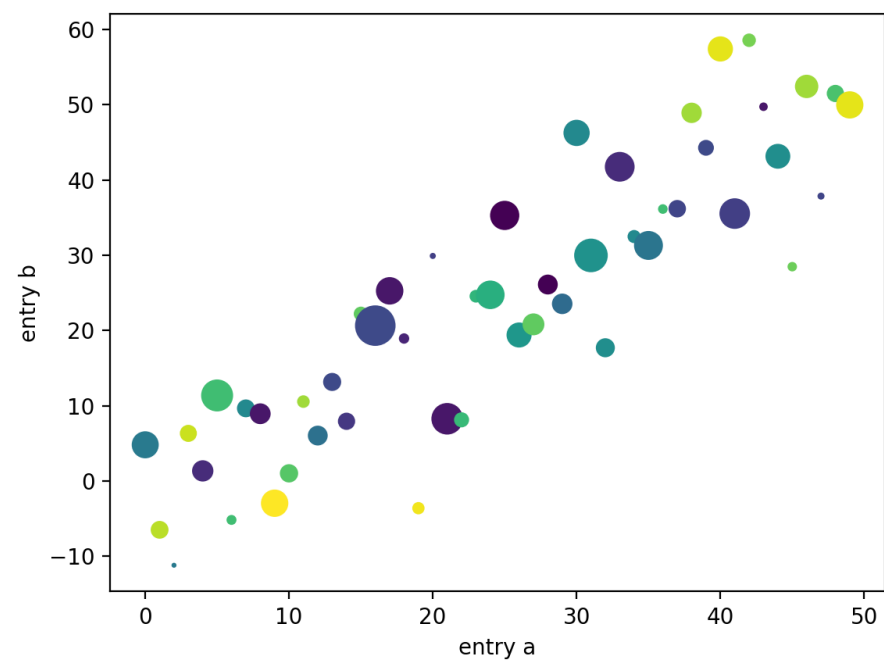
```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
np.random.seed(3)
x = 4 + np.random.normal(0, 2, 24)
y = 4 + np.random.normal(0, 2, len(x))
sizes = np.random.uniform(15, 80, len(x))
colors = np.random.uniform(15, 80, len(x))
fig, ax = plt.subplots()
ax.scatter(x, y, s=sizes, c=colors, vmin=0, vmax=100)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```



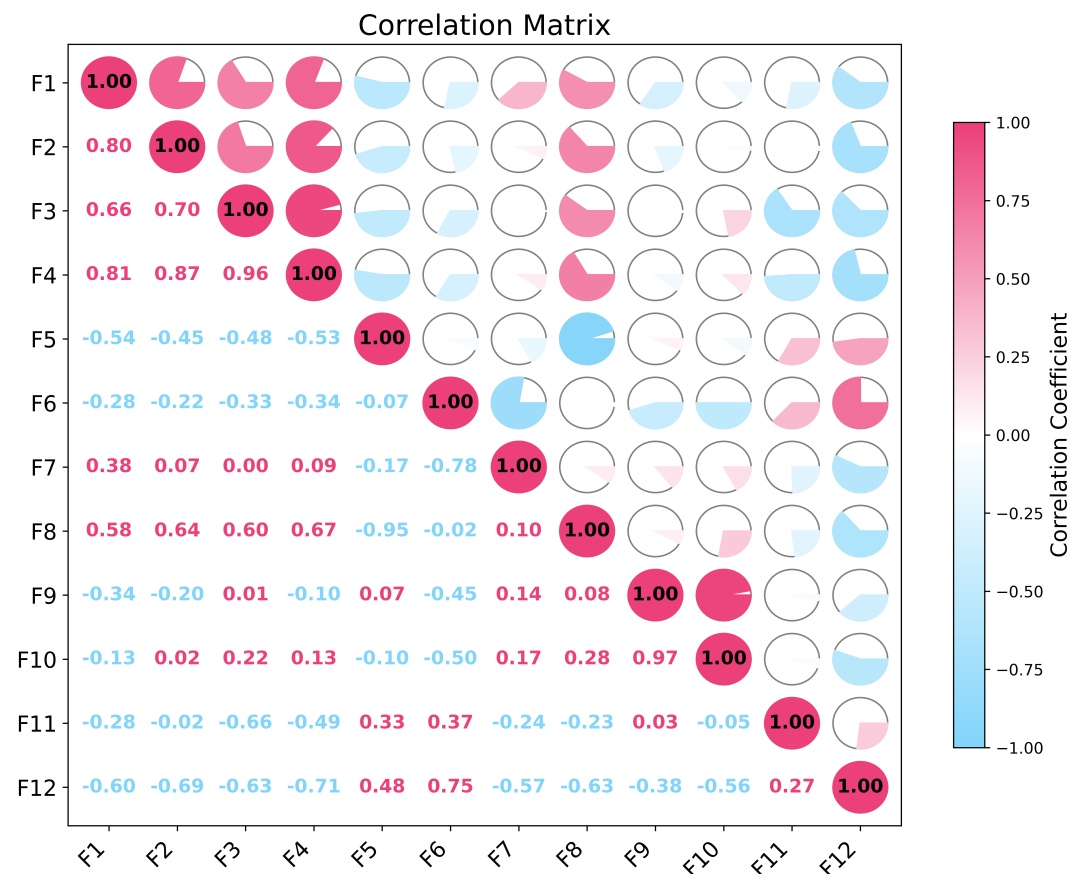
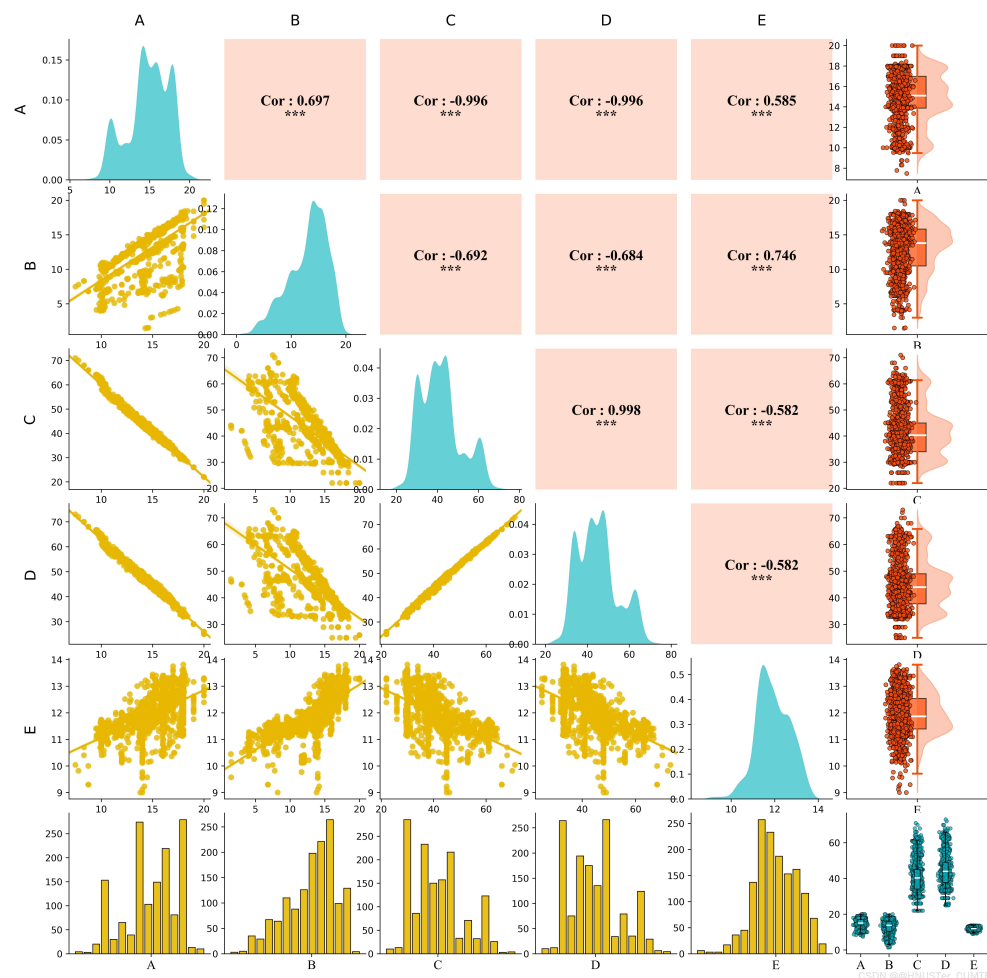
3D散点图

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
np.random.seed(19680801)
n = 100
rng = np.random.default_rng()
xs = rng.uniform(23, 32, n)
ys = rng.uniform(0, 100, n)
zs = rng.uniform(-50, -25, n)
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
ax.scatter(xs, ys, zs)
ax.set(xticklabels=[], yticklabels=[], zticklabels=[])
plt.show()
```



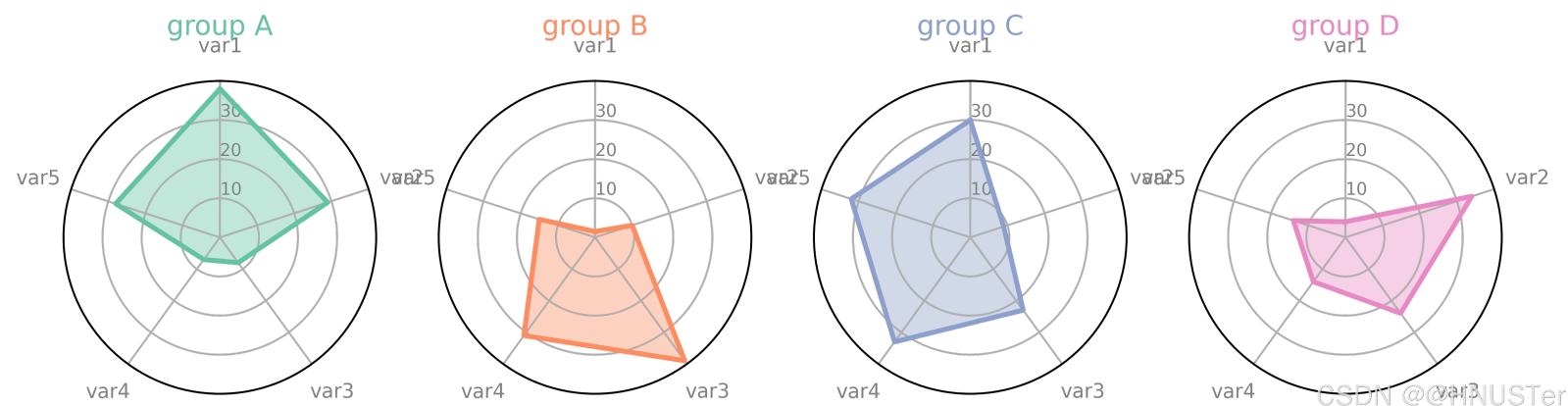
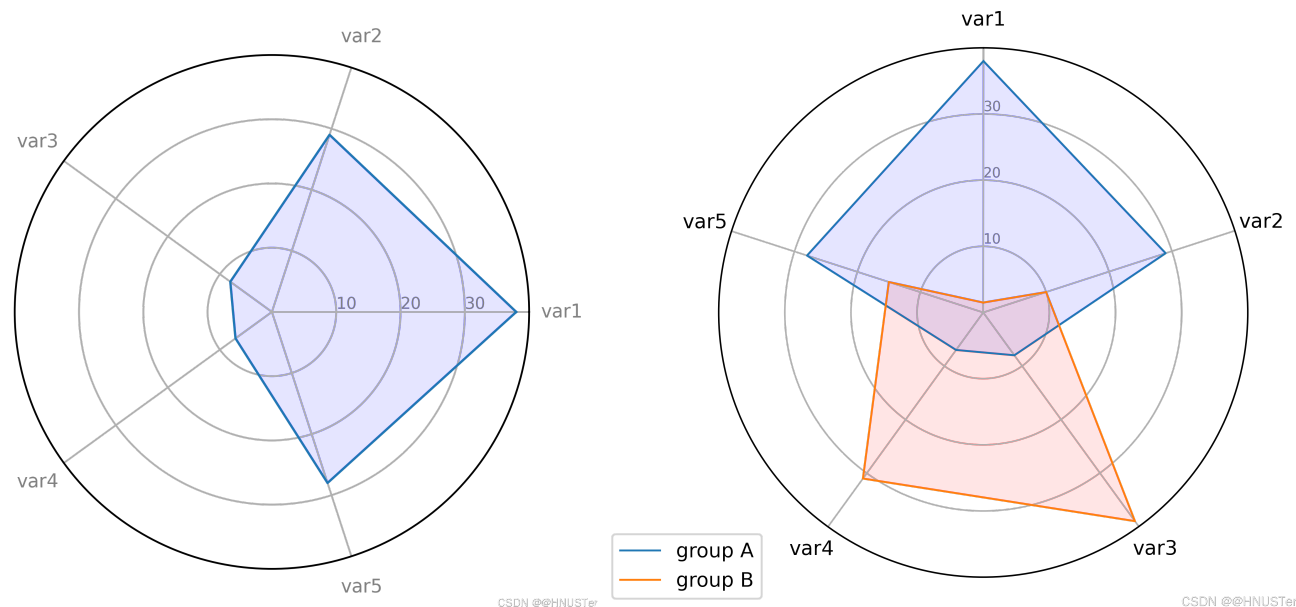


散点图矩阵和相关系数矩阵：Python数据可视化科技图表绘制系列教程（七）-CSDN博客 [↗](#)



CSDN @HNUTer_CUMTBer

雷达图：Python数据可视化科技图表绘制系列教程（四）-CSDN博客 ↗



彩蛋：心形线 (Cardioid)

```
import numpy as np
theta= np.arange(0, 2*np.pi, 0.05)
r=5*(1-np.sin(theta))
ax = plt.subplot(111, projection='polar')
ax.plot(theta, r, lw=3, c='red')
ax.grid(True)
```

3. 描述性统计

3.1 描述水平的统计量

简单平均数

```
import pandas as pd
example4_1 = pd.read_csv("F:/pydata/example/chap04/example4_1.csv")
example4_1['分数'].mean() # 或写成pd.DataFrame.mean(example4_1)
```

加权平均数

```
import numpy as np
example4_2 = pd.read_csv("F:/pydata/example/chap04/example4_2.csv")
m = example4_2['组中值']
f = example4_2['人数']
x_bar = np.average(a = m, weights = f)
```

中位数

```
example4_1['分数'].median()
```

四分位数

```
np.quantile(example4_1['分数'], q = [0.25,0.75], interpolation = 'linear')
```

百分位数

```
np.quantile(example4_1['分数'], q = [0.1,0.25,0.5,0.75,0.9], interpolation = 'linear')
```

众数

```
example4_1['分数'].mode()
```

```
import scipy  
from scipy import stats  
mode = stats.mode(example4_1['分数'])
```

3.2 描述差异的统计量

极差

```
R = example4_1['分数'].max() - example4_1['分数'].min()
```

四分位差

```
IQR = np.quantile(example4_1['分数'], q = 0.75) - np.quantile(example4_1['分数'], q = 0.25)
```

方差

```
example4_1['分数'].var()
```

标准差

```
example4_1['分数'].std()
```

加权标准差

```
m = example4_2['组中值']; f = example4_2['人数']  
wm = np.average(a = m, weights = f)  
var = sum(((m - wm)**2)*f) / (sum(f) - 1)  
std = var**(1/2)
```

变异系数

```
example4_10 = pd.read_csv("F:/pydata/example/chap04/example4_10.csv")  
mean = example4_10.mean()  
std = example4_10.std()  
cv = std / mean  
np.round(cv,4) # 保留四位小数
```

3.3 描述分布形状的统计量

偏度系数

```
example4_1['分数'].skew()
```

峰度系数

```
example4_1['分数'].kurt()
```

3.4 数据标准化

标准分数

```
z = stats.zscore(example4_1['分数'], ddof = 1) # ddof是自由度
```

极值标准化

```
x = example4_1['分数']  
T = (x - min(x))/(max(x) - min(x))
```

4. 课后习题

习题1（数据可视化）

`exercise3_3.csv`（该数据集来自R语言的`mtcars`）摘自1974年《美国汽车趋势》杂志，包括32款汽车（1973-1974年款）的油耗、汽车设计和性能等共11个变量。根据该数据集绘制以下图形。

- (1) 绘制气缸数量(`cyl`)的直方图，并为直方图叠加核密度曲线。
- (2) 绘制每加仑油行驶的英里数(`mpg`)和汽车自重(`wt`)两个变量的箱线图和小提琴图。
- (3) 绘制该数据集的散点图矩阵和相关系数矩阵图。
- (4) 绘制每加仑油行驶的英里数(`mpg`)、总马力(`hp`)和汽车自重(`wt`)3个变量的3D散点图和气泡图。

习题2（描述性统计）

随机抽取50个网络购物的消费者，调查他们某月的网购金额（单位：元），结果见`exercise4_1.csv`。

- (1) 计算平均数、标准差、极差和四分位差。
- (2) 计算10%、50%、75%的分位数。
- (3) 计算偏度系数和峰度系数，分析网购金额的分布特征。
- (4) 计算标准分数和极值标准化值，检测数据的离群点。

作业要求

- 提交代码+对应图形+必要解释，注意标明题号
- 整理成一个pdf文档，命名为学号-姓名如“2410000-张三”
- **DDL**：10月16日24点前发送至助教飞书（2120253538）
- 视代码完整性、作图美观性、答案准确性评分
- 逾期提交、抄袭雷同记0分
- 可以使用AI

欢迎交流 ~



zzynankai@outlook.com



Bilibili: 西山yu



xishanyu2.github.io