第5章 数据可视化

学习目标

- 使用matplotlib和seaborn模块绘制折线图、直方图、柱状图、饼图、散点图等图表
- 掌握如何设置并修改图表信息
- 使用PyEcharts模块实现地理数据可视化

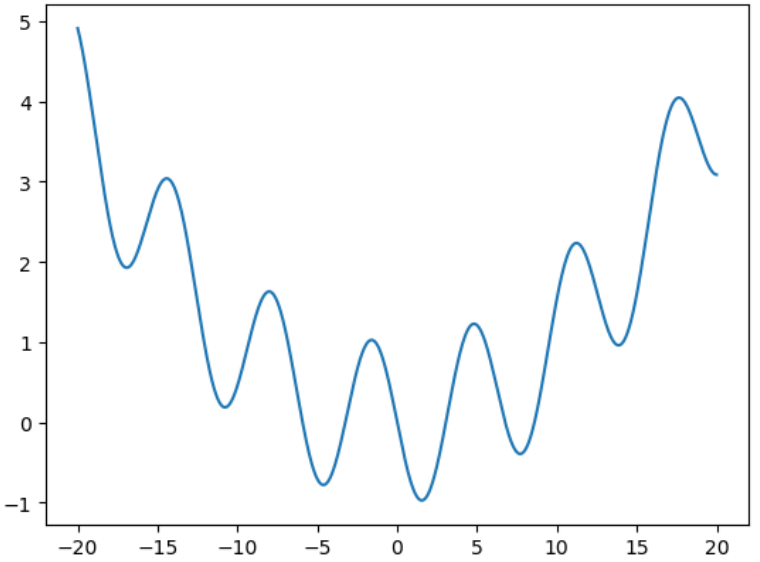
5.1 Matplotlib

- Matplotlib是Python中最常用的可视化工具之一。
- Matplotlib可以方便地创建海量类型的二维图表和基本的三维图表，并且还集成了方便快捷的绘图模块。

5.1.1 图表的创建

- 创建二维图表的基本方式是使用pyplot模块的plot()函数。
  - 该函数需要传入两个长度一致的横纵坐标数组，横纵坐标数组对应位置上的值构成一个点，plot()会将这些点放在图像中。在未指定其他参数的情况下默认会将所有点连线。
  - 在使用plot()构造图像后，还需要调用show()函数将图像展示出来。

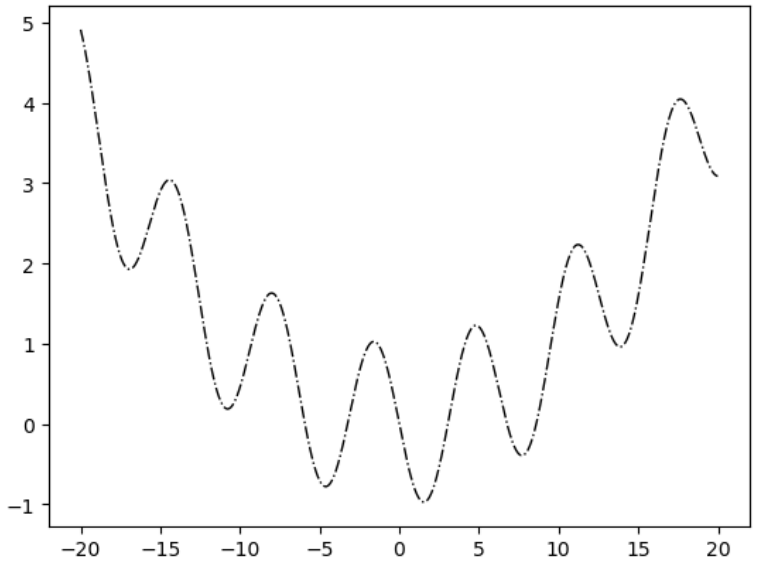
```
from matplotlib import pyplot as plt
import numpy as np

def fun(x):
    return 0.01*x**2-np.sin(x)

x = np.arange(-20.0, 20.0, 0.01)
y = fun(x)
plt.plot(x, y)
plt.savefig('test.jpg')
plt.show()
```



5.1.1 图表的创建

- plot()函数含有几个常用参数：
  - color：可以设置线条颜色。支持常用颜色的英文名称，如'blue'；也支持颜色十六进制值，如'#DC143C'。
  - linestyle：设置线型。支持的线型有'-'、'--'、'-.'、':'、'None'、'solid'、'dashed'、'dashdot'、'dotted'等。
  - linewidth：设置线条宽度，默认值为1.5。

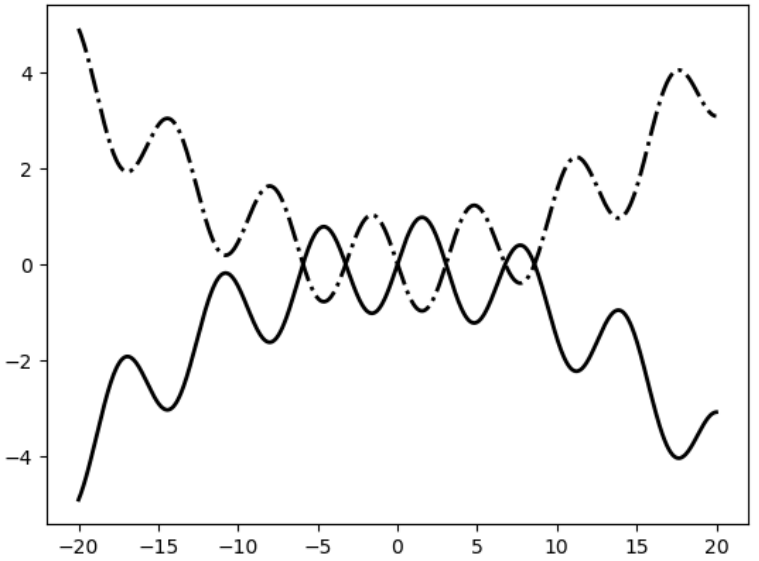
```
plt.plot(x, y, color = 'black', linestyle = '-.', linewidth = 1)
plt.show()
```



5.1.2 绘制多函数图像

- 将多条曲线绘制在同一个坐标轴中可以通过多次调用plot()函数实现：

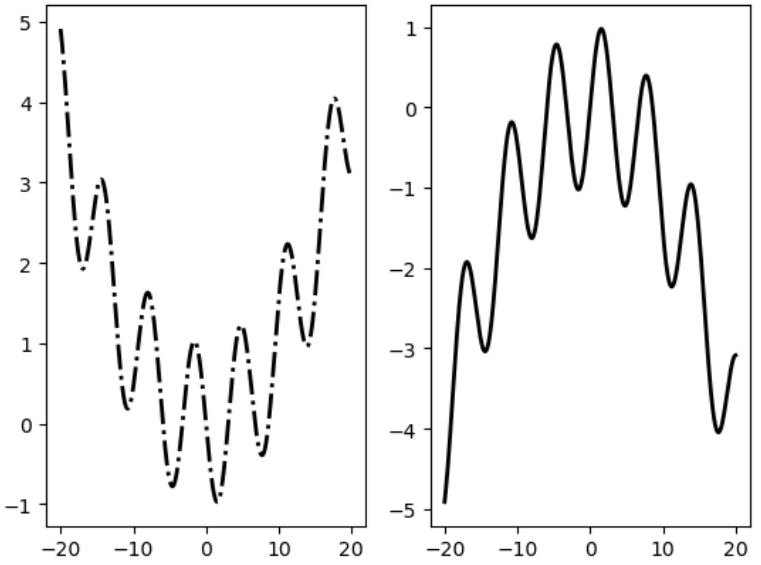
```
from matplotlib import pyplot as plt
import numpy as np

def fun(x):
    return 0.01*x**2-np.sin(x)

x = np.arange(-20.0, 20.0, 0.01)
y = fun(x)
plt.plot(x, y, color = 'black', linestyle = '-.', linewidth = 2)
plt.plot(x, -y, color = 'black', linestyle = '-', linewidth = 2)
plt.show()
```



5.1.2 绘制多函数图像

- 同时绘制多个图像并列显示，即子图，可以通过pyplot.subplot()函数实现：
  - 首先，利用pyplot.figure()函数创建一张新图，表示之后的操作是基于该图像的操作；
  - 其次，利用pyplot.subplot()函数创建一张子图，其参数为三位正整数，分别表示行数、列数、和索引值；
  - 然后，利用pyplot.plot()函数画出所需要画的图形。

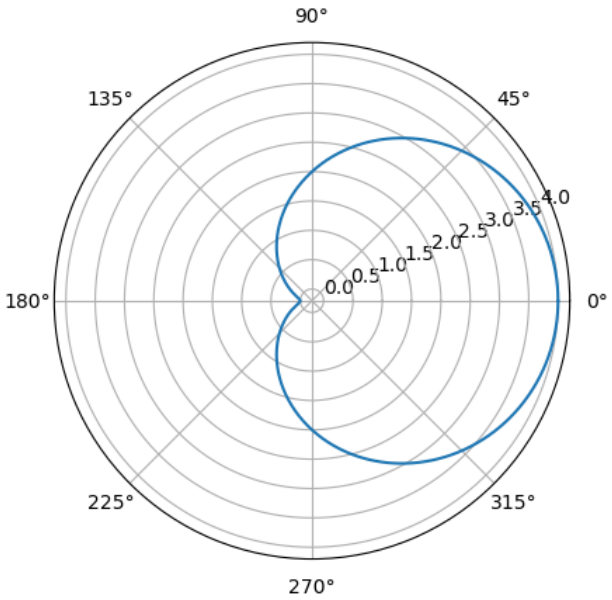
```
def fun(x):
    return 0.01*x**2-np.sin(x)

x = np.arange(-20.0, 20.0, 0.01)
y = fun(x)
plt.figure(1)
plt.subplot(121)
plt.plot(x, y, color = 'black', linestyle = '-.', linewidth = 2)
plt.subplot(122)
plt.plot(x, -y, color = 'black', linestyle = '-', linewidth = 2)
plt.show()
```



5.1.2 绘制多函数图像

- pyplot.subplot()函数还有很多其他功能，如在子图状态下定义极坐标系，画出极坐标系下的图形:

```
a = np.arange(0, 2*np.pi, 0.01)
fig = plt.figure(1)
ax1 = plt.subplot(111, projection = 'polar')
ax1.plot(a, 2*(1+np.cos(a)))
plt.show()
```



5.1.3 添加图表信息

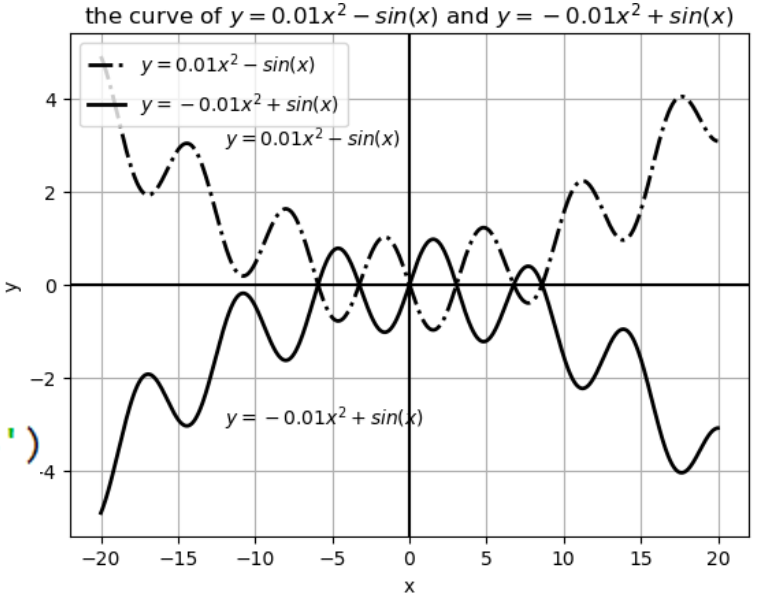
- pyplot模块提供丰富的方法设置和修改图表信息：

函数	说明
pyplot.axis()	设置图表显示的坐标轴范围
pyplot.xlim()	设置图表横坐标范围
pyplot.ylim()	设置图表纵坐标范围
pyplot.xlabel()	设置图表横坐标的标签
pyplot.ylabel()	设置图表纵坐标的标签
pyplot.title()	设置图表的标题
pyplot.legend()	设置图表的图例信息
pyplot.grid()	设置图表网格线
pyplot.axhline()	添加水平直线
pyplot.axvline()	添加垂直直线
pyplot.text()	添加文本
pyplot.annotate()	添加注释

5.1.3 添加图表信息

- 添加横纵坐标的标签、图表的标题、图例、文本和网格线等:

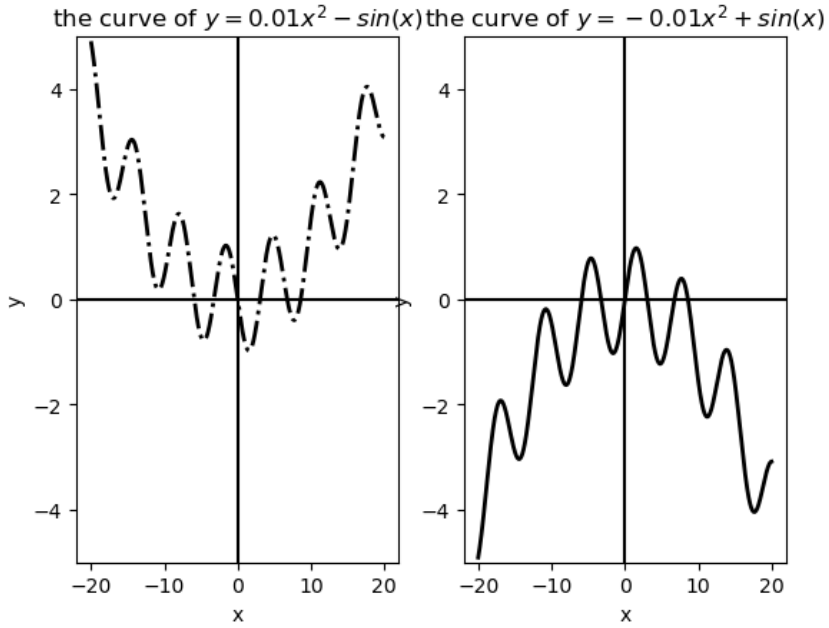
```
def fun(x):
    return 0.01*x**2-np.sin(x)

x = np.arange(-20.0, 20.0, 0.01)
y = fun(x)
plt.plot(x, y, color = 'black', linestyle = '-.', linewidth = 2)
plt.plot(x, -y, color = 'black', linestyle = '-', linewidth = 2)
plt.xlabel('x')
plt.ylabel('y')
plt.title('the curve of $y=0.01x^{2}-sin(x)$ and $y=-0.01x^{2}+sin(x)$')
plt.legend(['$y=0.01x^{2}-sin(x)$', '$y=-0.01x^{2}+sin(x)$'], loc = 2)
plt.axvline(x = 0, color = 'black')
plt.axhline(y = 0, color = 'black')
plt.text(-12, 3, '$y=0.01x^{2}-sin(x)$')
plt.text(-12, -3, '$y=-0.01x^{2}+sin(x)$')
plt.grid()
plt.show()
```



5.1.3 添加图表信息

- 将两张子图并列显示：

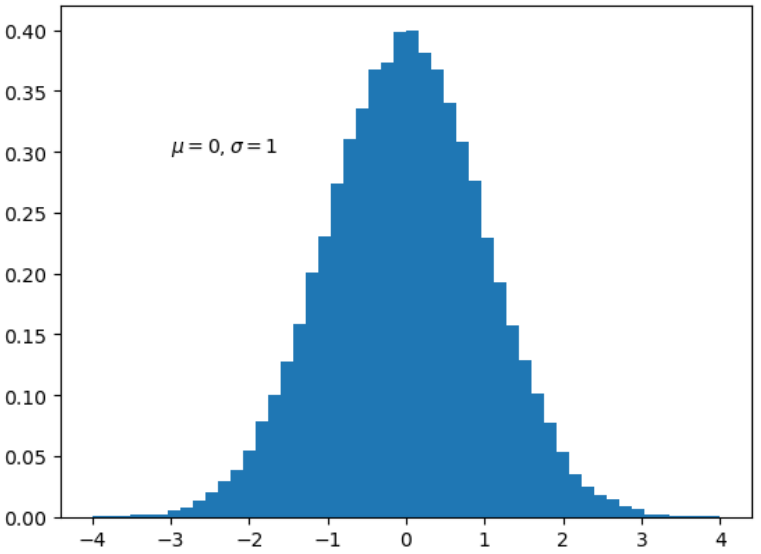
```
plt.figure(1)
plt.subplot(121)
plt.plot(x, y, color = 'black', linestyle = '-.', linewidth = 2)
plt.ylim(-5, 5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('the curve of $y=0.01x^{2}-sin(x)$')
plt.axvline(x = 0, color = 'black')
plt.axhline(y = 0, color = 'black')
plt.subplot(122)
plt.plot(x, -y, color = 'black', linestyle = '-', linewidth = 2)
plt.ylim(-5, 5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('the curve of $y=-0.01x^{2}+sin(x)$')
plt.axvline(x = 0, color = 'black')
plt.axhline(y = 0, color = 'black')
plt.subplot(122)
plt.show()
```



5.1.4 不同类型的图表

- 除了之前介绍的曲线图，pyplot模块还支持其他类型的基本图形：
- 直方图：pyplot.hist()
  - 直方图能够直观地显示各组频数/频率的分布情况和各组之间的差别。
  - 可以使用pyplot.hist()来绘制直方图，该函数有几个重要参数：
    - x：用于绘制直方图的数据，支持列表和NumPy数组；
    - bins：直方图中箱子的数量；
    - range：图形的上下限，以列表形式传入；
    - normed：0为频数分布直方图，1为频率分布直方图，默认为0。

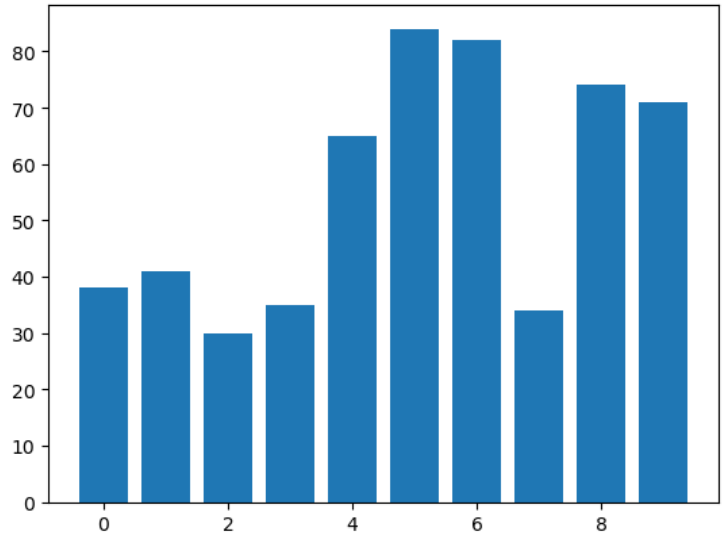
```
data = np.random.randn(50000)
plt.hist(x = data, bins = 50, range = [-4, 4], normed = 1)
plt.text(-3, 0.3, r'$\mu=0, \sigma = 1$')
plt.show()
```



5.1.4 不同类型的图表

- 柱状图：pyplot.bar()
  - 柱状图也称条形图，能够清晰地揭示各组数据的大小，便于比较各组间数据的差别。
  - 可以通过pyplot.bar()呈现，它的两个主要参数x和height，分别表示横坐标和对应柱子的高度。

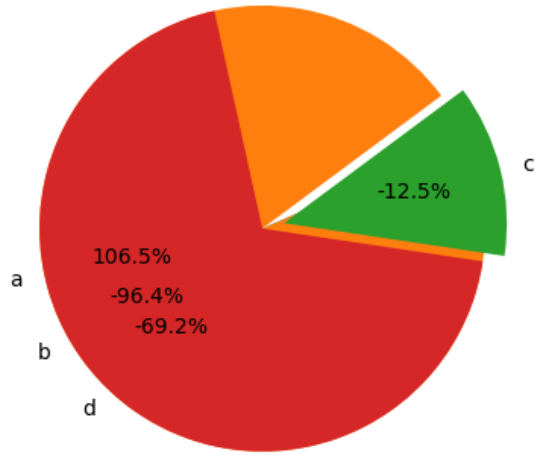
```
x = np.arange(0, 10)
y = [i for i in np.random.randint(20, 100, 10)]
plt.bar(x = x, height = y)
plt.show()
```



5.1.4 不同类型的图表

- 饼图：pyplot.pie()
  - 饼图可以直观地展示总体中各组成部分所占的比重，通过pyplot.pie()可以绘制饼图。
  - 该函数的重要参数有：
    - x：类型为列表或NumPy数组等的数据；
    - labels：数据标签；
    - explode：离中心的距离；
    - autopct：控制饼图内百分比设置；
    - radius：控制饼图半径，默认值为1。

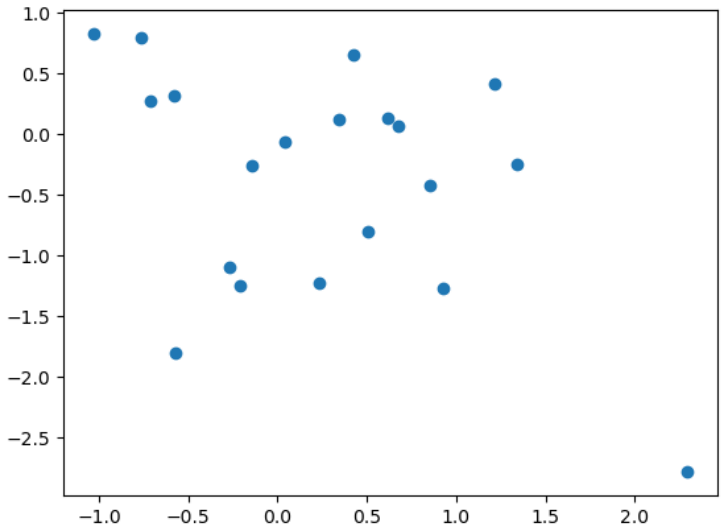
```
data = np.random.randn(4)
lable = ['a', 'b', 'c', 'd']
explode = [0, 0, 0.1, 0]
plt.pie(x = data, labels = lable, explode = explode, autopct = '%2.1f%%')
plt.show()
```



5.1.4 不同类型的图表

- 散点图：pyplot.scatter()
  - 散点图既可以清晰地展示数据点的分布情况，也可以发现变量之间的关系。
  - 可以通过pyplot.scatter()函数进行绘制，其两个参数x和y，分别表示横纵坐标。

```
x = np.random.randn(20)
y = np.random.randn(20)
plt.scatter(x, y)
plt.show()
```



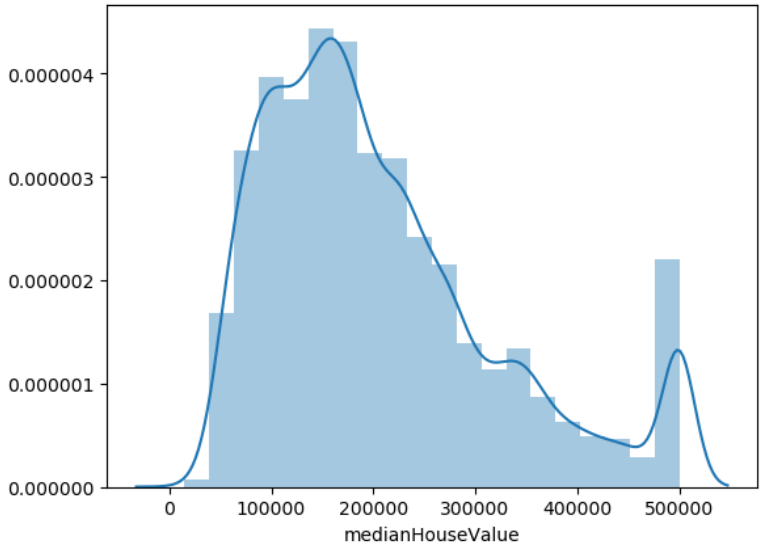
5.2 Seaborn

- Seaborn是基于Matplotlib的图形可视化库。它提供了一种高度交互式界面，用户能够做出各种有吸引力的统计图表。
- Seaborn能高度兼容Numpy、Series和DataFrame等数据结构以及scipy与statsmodels等统计模式的可视化。

5.2.1 直方图

- 使用Seaborn库绘制直方图的函数是seaborn.distplot()，该函数的主要参数有：
  - a：数据列，支持多种数据类型；
  - bins：直方图中箱子的数量；
  - kde：False表示不显示核密度估计，显示频数分布直方图，True表示显示核密度估计，显示频率分布直方图，默认为True。

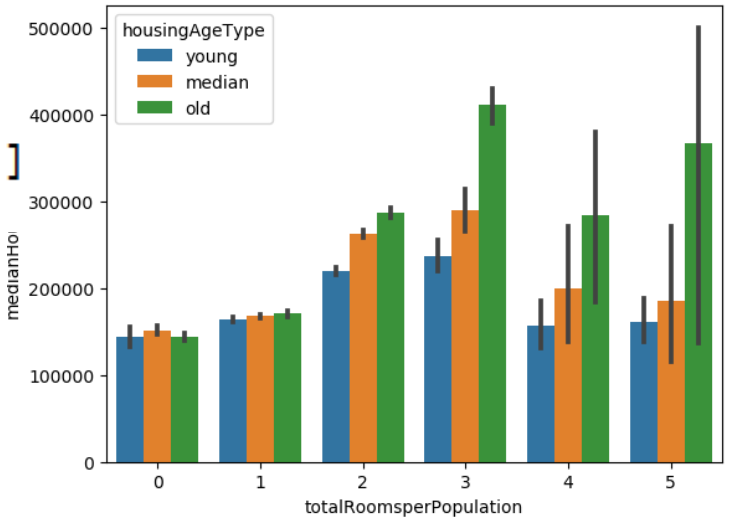
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

data = pd.read_csv('./cal_housing.data')
data.columns = ['longitude', 'latitude',
                'housingMedianAge','totalRooms',
                'totalBedrooms', 'population', 'households',
                'medianIncome', 'medianHouseValue']
sns.distplot(data['medianHouseValue'], bins = 20, kde = True)
plt.show()
```



5.2.2 柱状图

- seaborn.barplot()可以用来绘制柱状图，该函数的主要参数有：
  - x和y分别是横纵坐标数据，hue为分类变量，这3个参数可以是NumPy数组，也可以是Series，若data参数传入了DataFrame，该参数可只传入DataFrame的列索引；
  - data是数据集，若没有传入数据集，则前3个参数x、y和hue不能只传入列索引。

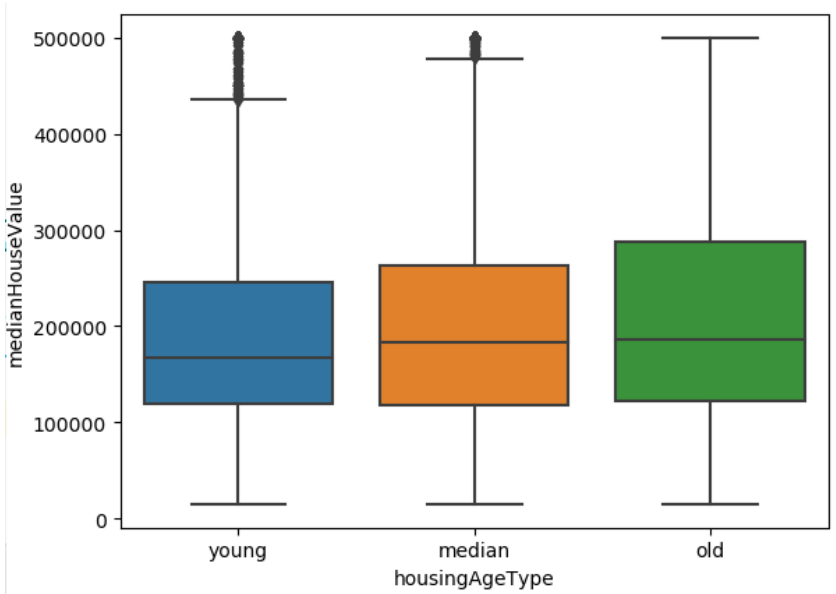
```
data['housingAgeType'] = pd.cut(data['housingMedianAge'],
                                bins = 3,
                                labels = ['young', 'median', 'old'])
data['totalRoomsperPopulation'] = data['totalRooms']/data['population']
data = data.astype({'totalRoomsperPopulation':'int'})
sns.barplot(x = 'totalRoomsperPopulation',
            y = 'medianHouseValue',
            hue = 'housingAgeType',
            data = data[data['totalRoomsperPopulation'] <= 5])
plt.show()
```



5.2.3 箱线图

- 绘制箱线图的函数是seaborn.boxplot()，该函数的主要参数与seaborn.barplot()相似。
  - 每个箱线都有5条线，从上往下依次为该类别的上边缘、75分位数、中位数、25分位数和下边缘。

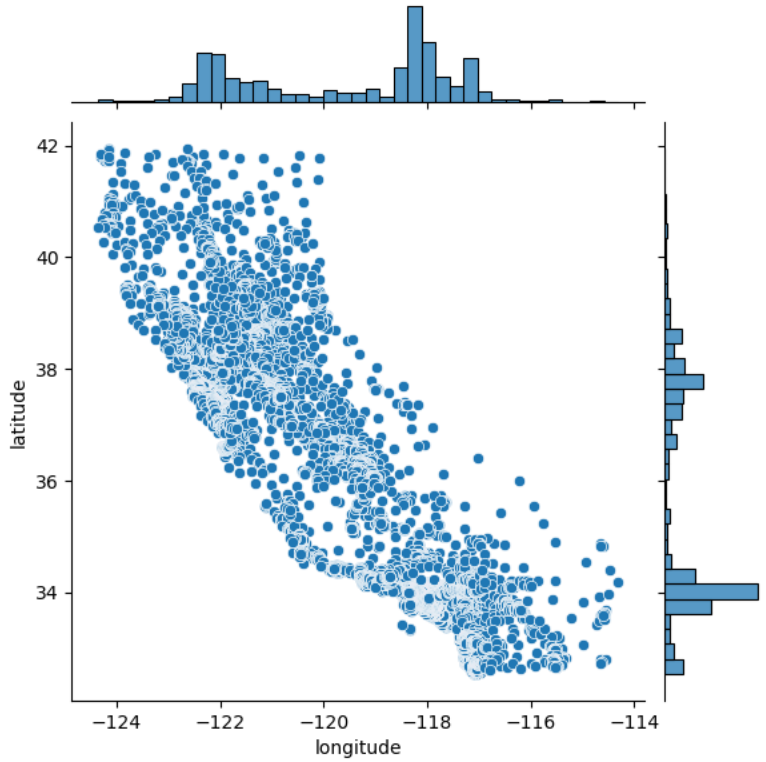
```
sns.boxplot(x = 'housingAgeType',y = 'medianHouseValue',data = data)
plt.show()
```



5.2.4 散点图

- 散点图的函数是seaborn.jointplot()，参数与seaborn.barplot()相似，但是没有hue参数。

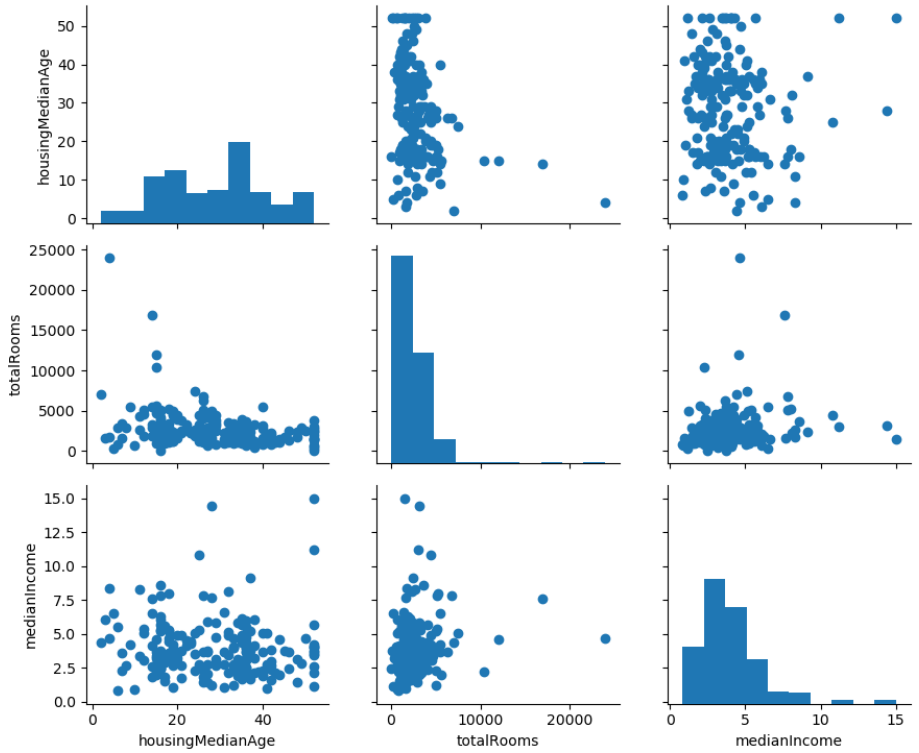
```
sns.jointplot(x = 'longitude', y = 'latitude', data = data)
plt.show()
```



5.2.5 结构化多图网格

- seaborn.PairGrid()可以绘制结构化多图网格，以快速提取有关复杂数据的大量信息。

```
fig = sns.PairGrid(data[['housingMedianAge',
                         'totalRooms',
                         'medianIncome']]
                   .sample(200, random_state = 20))
fig.map_diag(plt.hist)
fig.map_offdiag(plt.scatter)
plt.show()
```



5.2.6 回归图

- 使用回归图seaborn.lmplot()可以拟合两者的曲线，该函数参数与seaborn.barplot()相似。

```
sns.lmplot(x = 'medianIncome',
           y = 'medianHouseValue',
           hue = 'housingAgeType',
           data = data.sample(200 , random_state = 20),
           legend = False)
plt.legend(loc = 2, title = 'medianHouseValue')
plt.show()
```