**Набор инструментов mplot3d**

**Теоретическая часть**

Набор mplot3d включен во все стандартные версии matplotlib и позволят расширить возможности создания трехмерных визуализаций данных. Если объект Figure выводится в отдельном окне, то можно вращать оси трехмерного представления с помощью мыши.

С этим пакетом продолжает использоваться объект Figure, но вместо объектов Axes определяется новый тип, Axes3D из этого набора. Поэтому нужно добавить один импорт в код.

from mpl\_toolkits.mplot3d import Axes3D

### Трехмерные поверхности

Ранее для представления трехмерных поверхностей использовался контурный график. Но с помощью mplot3D поверхности можно рисовать в 3D.

Когда meshgrid вычислена, можно вывести поверхность графика с помощью функции plot\_surface().

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = Axes3D(fig)

X = np.arange(-2,2,0.1)

Y = np.arange(-2,2,0.1)

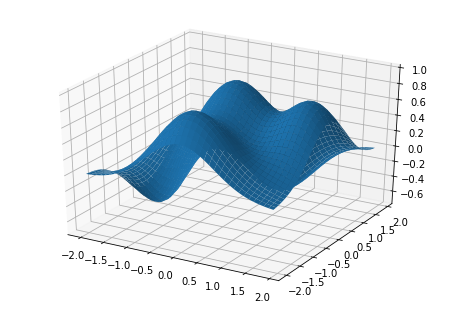
X,Y = np.meshgrid(X,Y)

def f(x,y):

return (1 - y\*\*5 + x\*\*5)\*np.exp(-x\*\*2-y\*\*2)

ax.plot\_surface(X,Y,f(X,Y), rstride=1, cstride=1)

plt.show()



Трехмерная поверхность выделяется за счет изменения карты с помощью именованного аргумента cmap. Поверхность также можно вращать с помощью функции view\_unit(). На самом деле, эта функция подстраивает точку обзора, откуда можно будет рассмотреть поверхность, изменяя аргументы elev и azim. С помощью их комбинирования можно получить поверхность, изображенную с любого угла. Первый аргумент настраивает высоту, а второй — угол поворота поверхности.

Например, можно поменять карту с помощью plt.cm.hot и повернуть угол на elev=30 и azim=125.

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = Axes3D(fig)

X = np.arange(-2,2,0.1)

Y = np.arange(-2,2,0.1)

X,Y = np.meshgrid(X,Y)

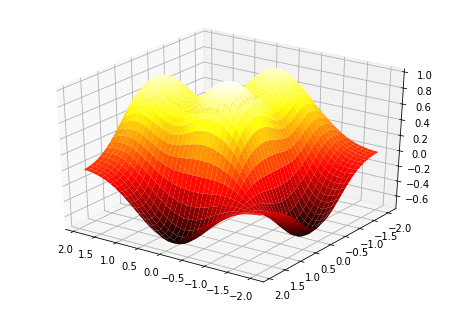
def f(x,y):

return (1 - y\*\*5 + x\*\*5)\*np.exp(-x\*\*2-y\*\*2)

ax.plot\_surface(X,Y,f(X,Y), rstride=1, cstride=1, cmap=plt.cm.hot)

ax.view\_init(elev=30,azim=125)

plt.show()



### Диаграмма рассеяния

Самым используемым трехмерным графиком остается 3D график рассеяния. С его помощью можно определить, следуют ли точки определенным трендам и, что самое важное, скапливаются ли они.

В этом случае используется функция scatter() как и при построении обычного двумерного графика, но на объекте Axed3D. Таким образом можно визуализировать разные объекты Series, представленные через вызовы функции scatter() в единой трехмерной форме.

from mpl\_toolkits.mplot3d import Axes3D

xs = np.random.randint(30,40,100)

ys = np.random.randint(20,30,100)

zs = np.random.randint(10,20,100)

xs2 = np.random.randint(50,60,100)

ys2 = np.random.randint(30,40,100)

zs2 = np.random.randint(50,70,100)

xs3 = np.random.randint(10,30,100)

ys3 = np.random.randint(40,50,100)

zs3 = np.random.randint(40,50,100)

fig = plt.figure()

ax = Axes3D(fig)

ax.scatter(xs,ys,zs)

ax.scatter(xs2,ys2,zs2,c='r',marker='^')

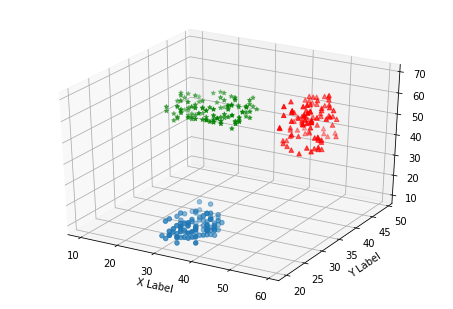
ax.scatter(xs3,ys3,zs3,c='g',marker='\*')

ax.set\_xlabel('X Label')

ax.set\_ylabel('Y Label')

ax.set\_xlabel('X Label')

plt.show()



## **Столбчатые диаграммы в 3D**

Также в анализе данных используются трехмерные столбчатые диаграммы. Здесь тоже нужно применять функцию bar() к объекту Axes3D. Если же определить несколько Series, то их можно накопить в нескольких вызовах функции bar() для одной 3D-визуализации.

from mpl\_toolkits.mplot3d import Axes3D

x = np.arange(8)

y = np.random.randint(0,10,8)

y2 = y + np.random.randint(0,3,8)

y3 = y2 + np.random.randint(0,3,8)

y4 = y3 + np.random.randint(0,3,8)

y5 = y4 + np.random.randint(0,3,8)

clr = ['#4bb2c5','#c5b47f','#EAA228','#579575','#839557','#958c12','#953579','#4b5de4']

fig = plt.figure()

ax = Axes3D(fig)

ax.bar(x,y,0,zdir='y',color=clr)

ax.bar(x,y2,10,zdir='y',color=clr)

ax.bar(x,y3,20,zdir='y',color=clr)

ax.bar(x,y4,30,zdir='y',color=clr)

ax.bar(x,y5,40,zdir='y',color=clr)

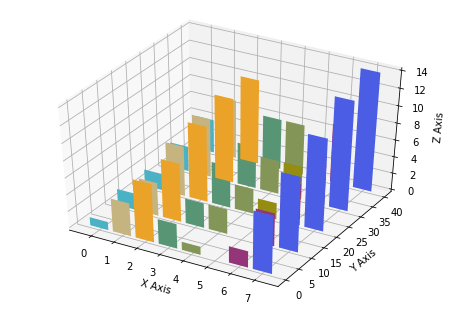
ax.set\_xlabel('X Axis')

ax.set\_ylabel('Y Axis')

ax.set\_zlabel('Z Axis')

ax.view\_init(elev=40)

plt.show()



## **Многопанельные графики**

Ранее были рассмотрены самые разные [способы представления данных на графике](https://pythonru.com/biblioteki/pyplot-uroki), в том числе и случаи, когда несколько наборов данных разделены в одном объекте Figure на нескольких подграфиках. В этом разделе речь пойдет о более сложных случаях.

### Подграфики внутри графиков

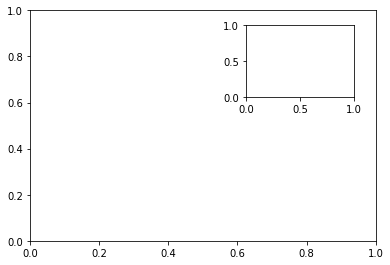
Есть возможность рассматривать графики внутри других ограниченных рамками. Поскольку речь идет о границах — объектах Axes — то есть необходимость разделять основные (основного графика) от тех, которые принадлежат добавляемому графику. Для этого используется функция figures(). С ее помощью нужно получить объект Figure, где будут определены два разных объекта Axes с помощью add\_axes().

fig = plt.figure()

ax = fig.add\_axes([0.1,0.1,0.8,0.8])

inner\_ax = fig.add\_axes([0.6,0.6,0.25,0.25])

plt.show()



Чтобы лучше понять эффект этого режима, лучше заполнить предыдущий объект Axes с помощью реальных данных.

fig = plt.figure()

ax = fig.add\_axes([0.1,0.1,0.8,0.8])

inner\_ax = fig.add\_axes([0.6,0.6,0.25,0.25])

x1 = np.arange(10)

y1 = np.array([1,2,7,1,5,2,4,2,3,1])

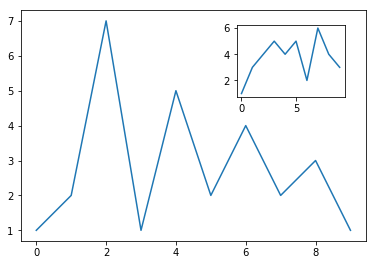
x2 = np.arange(10)

y2 = np.array([1,3,4,5,4,5,2,6,4,3])

ax.plot(x1,y1)

inner\_ax.plot(x2,y2)

plt.show()



### Сетка подграфиков

Есть другой способ создания подграфиков. С помощью функции subplots() их нужно добавить, разбив таким образом график на секторы. matplotlib позволяет работать даже с более сложными случаями с помощью функции GridSpec(). Это подразделение позволяет разбить область на сетку подграфиков, каждому из которых можно присвоить свой график, так что результатом будет сетка с подграфиками разных размеров с разными направлениями.

gs = plt.GridSpec(3,3)

fig = plt.figure(figsize=(6,6))

fig.add\_subplot(gs[1,:2])

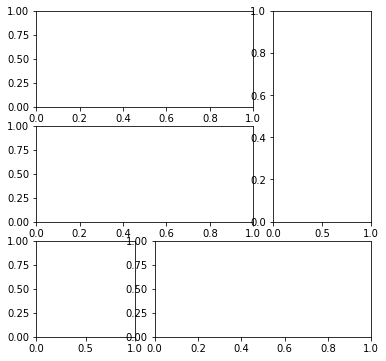
fig.add\_subplot(gs[0,:2])

fig.add\_subplot(gs[2,0])

fig.add\_subplot(gs[:2,2])

fig.add\_subplot(gs[2,1:])

plt.show()



Создать такую сетку несложно. Осталось разобраться, как заполнить ее данными. Для этого нужно использовать объект Axes, который возвращает каждая функция add\_subplot(). В конце остается вызвать plot() для вывода конкретного графика.

gs = plt.GridSpec(3,3)

fig = plt.figure(figsize=(6,6))

x1 = np.array([1,3,2,5])

y1 = np.array([4,3,7,2])

x2 = np.arange(5)

y2 = np.array([3,2,4,6,4])

s1 = fig.add\_subplot(gs[1,:2])

s1.plot(x,y,'r')

s2 = fig.add\_subplot(gs[0,:2])

s2.bar(x2,y2)

s3 = fig.add\_subplot(gs[2,0])

s3.barh(x2,y2,color='g')

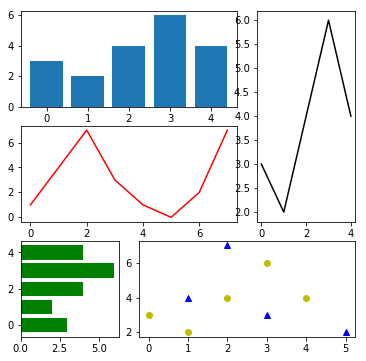
s4 = fig.add\_subplot(gs[:2,2])

s4.plot(x2,y2,'k')

s5 = fig.add\_subplot(gs[2,1:])

s5.plot(x1,y1,'b^',x2,y2,'yo')

plt.show()



**Задание (с использованием библиотеки mplot3d)**

1. Создайте график трехмерной поверхности

|  |  |
| --- | --- |
| **Четные варианты** | **Нечетные варианты** |
|  |  |
|  |  |

Поменяйте для графика цветовую карту, постройте графики для двух рахных углов зрения.

1. Постройте объемную диаграмму рассеяния на основании данных файла mlb\_players.csv по столбцам "Height(inches)", "Weight(lbs)", "Age".
2. Постройте трёхмерную столбчатую диаграмму для следующего набора данных:

|  |  |  |
| --- | --- | --- |
| X | Y | Z |
| 1 | 36 | 33 |
| 2 | 5 | 32 |
| 3 | 6 | 5 |
| 4 | 67 | 55 |
| 5 | 0 | 3 |
| 6 | 3 | 5 |
| 7 | 45 | 77 |
| 8 | 6 | 5 |
| 9 | 6 | 88 |
| 10 | 7 | 47 |

1. Постройте многопанельные графики
   1. Подграфик f(x)=cos(x) внутри графика f(x)=cos(x)+1/sin(x)\*x2
   2. Выберите любые 4 ранее построенных графика и отобразите их все одновременно с помощью функции GridSpec()