[python中matplotlib绘制图形](http://blog.csdn.net/u011367448/article/details/17370393)

[matplotlib](http://matplotlib.sourceforge.net/) 是python最著名的绘图库，它提供了一整套和matlab相似的命令API，十分适合交互式地进行制图。该包下面有很多对象，比如pylab，pyplot等等，pylab集成了pyplot和numpy两个模块，能够进行快速绘图。Pylab和pyplot (<http://matplotlib.org/api/pyplot_api.html>)都能够通过对象或者属性对图像进行操作。Pyplot下也有很多对象，如figure，Axes对象等等，对图像进行细节处理。下面通过一些简单和常用的例子进行操作。

(下文参考：<http://blog.csdn.net/u011367448/article/details/17370393>)

# 0、常用命令参数

plt.title 设置图标标题

plt.xlabel 设置x轴标签, plt.ylabel 设置y轴标签

plt.plot 绘制图像

plt.scatter 绘制点的图像

plt.tick\_params 设置刻度标记大小(which参数表示设置轴),包括显示字体

plt.axis 设置每个坐标轴的取值范围

plt.xlim 设置当前x轴的显示范围

plt.ylim 设置当前y轴的显示范围

plt.savefig 自动保存图表

plt.figure 设置图片属性，比如设置多个图以及绘制窗口的尺寸等

# 1、为项目设置matplotlib参数

无论你工作在什么项目上，IPython都是值得推荐的。利用ipython --pylab，可以进入PyLab模式，已经导入了matplotlib库与相关软件包（例如Numpy和Scipy)，额可以直接使用相关库的功能。

这样IPython配置为使用你所指定的matplotlib GUI后端（TK/wxPython/PyQt/Mac OS X native/GTK)。对于大部分用户而言，默认的后端就已经够用了。Pylab模式还会向IPython引入一大堆模块和函数以提供一种更接近MATLAB的界面。

## 1.1 matplotlib图标正常显示中文

为了在图表中能够显示中文和负号等，需要下面一段设置：

|  |  |
| --- | --- |
| 1  2  3 | import matplotlib.pyplot as plt  font = FontProperties(fname=r"C:\Windows\Fonts\simsun.ttc", size=15) #用来正常显示中文标签  plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号 |

## 1.2 matplotlib inline和pylab inline

可以使用ipython --pylab打开ipython命名窗口。

|  |  |
| --- | --- |
| 1  2 | %matplotlib inline #notebook模式下  %pylab inline #ipython模式下 |

这两个命令都可以在绘图时，将图片内嵌在交互窗口，而不是弹出一个图片窗口，但是，有一个缺陷：除非将代码一次执行，否则，无法叠加绘图，因为在这两种模式下，是要有plt出现，图片会立马show出来，因此：

**推荐在ipython notebook时使用，这样就能很方便的一次编辑完代码，绘图。**

在代码执行过程中，有两种方式更改参数：

* 使用参数字典(rcParams)
* 调用matplotlib.rc()命令 通过传入关键字元祖，修改参数

如果不想每次使用matplotlib时都在代码部分进行配置，可以修改matplotlib的文件参数。可以用matplot.get\_config()命令来找到当前用户的配置文件目录。

配置文件包括以下配置项：

axes: 设置坐标轴边界和表面的颜色、坐标刻度值大小和网格的显示

ax.axis([xmin, xmax, ymin, ymax]) #设置横-纵轴的刻度

backend: 设置目标暑促TkAgg和GTKAgg

figure: 控制dpi、边界颜色、图形大小、和子区( subplot)设置

font: 字体集（font family）、字体大小和样式设置

grid: 设置网格颜色和线性

legend: 设置图例和其中的文本的显示

line: 设置线条（颜色、线型、宽度等）和标记

patch: 是填充2D空间的图形对象，如多边形和圆。控制线宽、颜色和抗锯齿设置等。

savefig: 可以对保存的图形进行单独设置。例如，设置渲染的文件的背景为白色。

verbose: 设置matplotlib在执行期间信息输出，如silent、helpful、debug和debug-annoying。

xticks和yticks: 为x,y轴的主刻度和次刻度设置颜色、大小、方向，以及标签大小。

# 2、线条相关属性标记设置

## 2.1 用来表示线条的属性

|  |  |  |  |
| --- | --- | --- | --- |
| **线条风格linestyle或ls** | **描述** | **线条风格linestyle或ls** | **描述** |
| ‘-‘ | 实线 | ‘:’ | 虚线 |
| ‘–’ | 破折线 | ‘None’,’ ‘,’’ | 什么都不画 |
| ‘-.’ | 点划线 |  |  |

## 2.2 线条标记

|  |  |  |  |
| --- | --- | --- | --- |
| **标记maker** | **描述** | **标记** | **描述** |
| ‘o’ | 圆圈 | ‘.’ | 点 |
| ‘D’ | 菱形 | ‘s’ | 正方形 |
| ‘h’ | 六边形1 | ‘\*’ | 星号 |
| ‘H’ | 六边形2 | ‘d’ | 小菱形 |
| ‘\_’ | 水平线 | ‘v’ | 一角朝下的三角形 |
| ‘8’ | 八边形 | ‘<’ | 一角朝左的三角形 |
| ‘p’ | 五边形 | ‘>’ | 一角朝右的三角形 |
| ‘,’ | 像素 | ‘^’ | 一角朝上的三角形 |
| ‘+’ | 加号 | ‘\‘ | 竖线 |
| ‘None’,’’,’ ‘ | 无 | ‘x’ | X |

## 3、颜色

可以通过调用matplotlib.pyplot.colors()得到matplotlib支持的所有颜色。

|  |  |  |  |
| --- | --- | --- | --- |
| **别名** | **颜色** | **别名** | **颜色** |
| b | 蓝色 | g | 绿色 |
| r | 红色 | y | 黄色 |
| c | 青色 | k | 黑色 |
| m | 洋红色 | w | 白色 |

如果这两种颜色不够用，还可以通过两种其他方式来定义颜色值：

* 使用HTML十六进制字符串 color='eeefff' 使用合法的HTML颜色名字（’red’,’chartreuse’等）。
* 也可以传入一个归一化到[0,1]的RGB元祖。 color=(0.3,0.3,0.4)

很多方法可以介绍颜色参数，如title()。

plt.tilte('Title in a custom color',color='#123456'）

## 4、背景色

通过向如matplotlib.pyplot.axes()或者matplotlib.pyplot.subplot()这样的方法提供一个axisbg参数，可以指定坐标这的背景色。

subplot(111,axisbg=(0.1843,0.3098,0.3098)

该图像的代码及解释见“第三章 chacd.py”

这里有个范例：

import matplotlib.pyplot as plt

#通过对象的形式文本作图

#控制dpi、边界颜色、图形大小、和子区( subplot)设置 #设置图形的尺寸，单位为英尺

fig = plt.figure()

#subplot(numRows, numCols, plotNum),如：subplot(221)

fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')#figure的标题

ax = fig.add\_subplot(111)   #figure下的坐标图

axfig.subplots\_adjust(top=0.85)

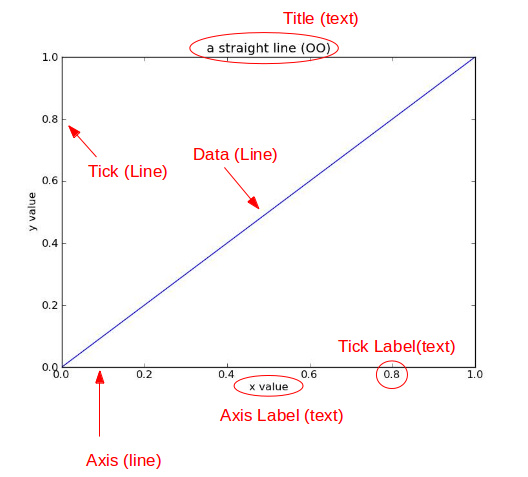
ax.set\_title('axes title')  #给ax设置标题

ax.set\_xlabel('xlabel')     #给ax设置横坐标

ax.set\_ylabel('ylabel')     #给ax设置纵坐标

## 5、基础（确定坐标值范围）

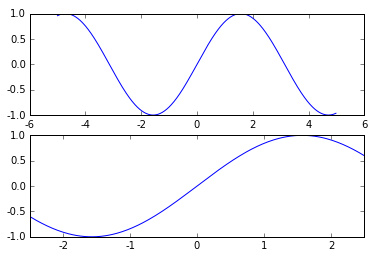
如果你向plot()指令提供了一维的数组或列表，那么matplotlib将默认它是一系列的y值，并自动为你生成x的值。默认的x向量从0开始并且具有和y同样的长度，因此x的数据是[0,1,2,3].



## 确定坐标范围

* **plt.axis([xmin, xmax, ymin, ymax])**
* 上面例子里的axis()命令给定了坐标范围。
* **xlim(xmin, xmax)和ylim(ymin, ymax)来调整x,y坐标范围**
* **在pyplot中使用plt.axis([xmin,xmax,ymin,ymax])来调整x,y坐标范围**

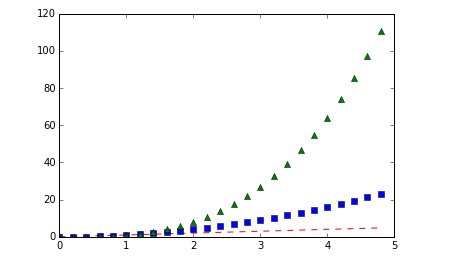
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | %matplotlib inline  import numpy as np  import matplotlib.pyplot as plt  from pylab import \*  x = np.arange(-5.0, 5.0, 0.02)  y1 = np.sin(x)  plt.figure(1)  plt.subplot(211)  plt.plot(x, y1)  plt.subplot(212)  #设置x轴范围  plt.xlim(-2.5, 2.5)  #设置y轴范围  plt.ylim(-1, 1)  plt.plot(x, y1) |



## 6、叠加图

用一条指令画多条不同格式的线。

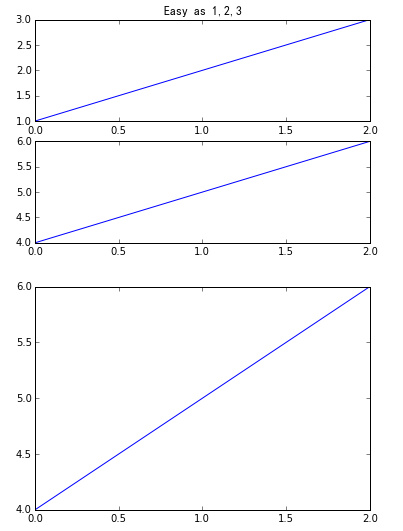
|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7** | **import numpy as np**  **import matplotlib.pyplot as plt**  **# evenly sampled time at 200ms intervals**  **t = np.arange(0., 5., 0.2)**  **# red dashes, blue squares and green triangles**  **plt.plot(t, t, 'r--', t, t\*\*2, 'bs', t, t\*\*3, 'g^')**  **plt.show()** |



## 7、plt.figure()

你可以多次使用figure命令来产生多个图，其中，图片号按顺序增加。这里，要注意一个概念当前图和当前坐标。所有绘图操作仅对当前图和当前坐标有效。通常，你并不需要考虑这些事，下面的这个例子为大家演示这一细节。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import matplotlib.pyplot as plt  plt.figure(1) # 第一张图  plt.subplot(211) # 第一张图中的第一张子图  plt.plot([1,2,3])  plt.subplot(212) # 第一张图中的第二张子图  plt.plot([4,5,6])  plt.figure(2) # 第二张图  plt.plot([4,5,6]) # 默认创建子图subplot(111)  plt.figure(1) # 切换到figure 1 ; 子图subplot(212)仍旧是当前图  plt.subplot(211) # 令子图subplot(211)成为figure1的当前图  plt.title('Easy as 1,2,3') # 添加subplot 211 的标题 |

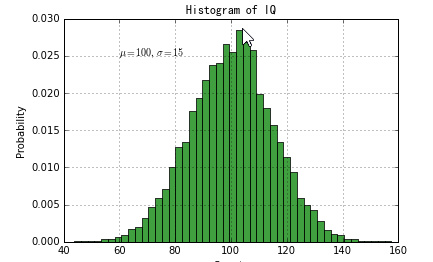


figure感觉就是给图像ID，之后可以索引定位到它。

## 8、plt.text()添加文字说明

* text()可以在图中的任意位置添加文字，并支持LaTex语法
* xlable(), ylable()用于添加x轴和y轴标签
* title()用于添加图的题目

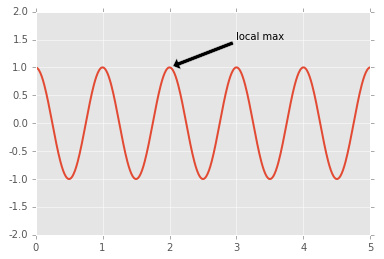
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import numpy as np  import matplotlib.pyplot as plt  mu, sigma = 100, 15  x = mu + sigma \* np.random.randn(10000)  # 数据的直方图  n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)  plt.xlabel('Smarts')  plt.ylabel('Probability')  #添加标题  plt.title('Histogram of IQ')  #添加文字  plt.text(60, .025, r'$\mu=100,\ \sigma=15$')  plt.axis([40, 160, 0, 0.03])  plt.grid(True)  plt.show() |



## 9、plt.annotate()文本注释

在数据可视化的过程中，图片中的文字经常被用来注释图中的一些特征。使用annotate()方法可以很方便地添加此类注释。在使用annotate时，要考虑两个点的坐标：被注释的地方xy(x, y)和插入文本的地方xytext(x, y)。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | import numpy as np  import matplotlib.pyplot as plt  ax = plt.subplot(111)  t = np.arange(0.0, 5.0, 0.01)  s = np.cos(2\*np.pi\*t)  line, = plt.plot(t, s, lw=2)  plt.annotate('local max', xy=(2, 1), xytext=(3,1.5),  arrowprops=dict(facecolor='black', shrink=0.05) )  plt.ylim(-2,2)  plt.show() |

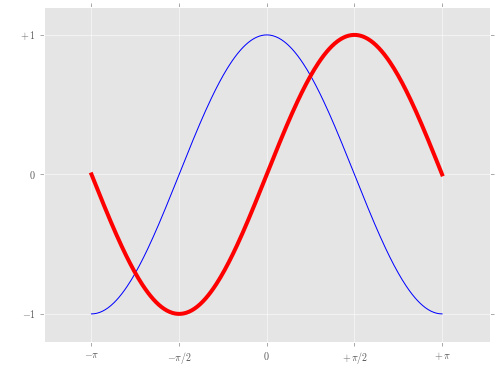


## 10、plt.xticks()/plt.yticks()设置轴记号

现在是明白干嘛用的了，就是人为设置坐标轴的刻度显示的值。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # 导入 matplotlib 的所有内容（nympy 可以用 np 这个名字来使用）  from pylab import \*  # 创建一个 8 \* 6 点（point）的图，并设置分辨率为 80  plt.figure(figsize=(8,6), dpi=80)  # 创建一个新的 1 \* 1 的子图，接下来的图样绘制在其中的第 1 块（也是唯一的一块）  plt.subplot(1,1,1)  X = np.linspace(-np.pi, np.pi, 256,endpoint=True)  C,S = np.cos(X), np.sin(X)  # 绘制余弦曲线，使用蓝色的、连续的、宽度为 1 （像素）的线条  plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")  # 绘制正弦曲线，使用绿色的、连续的、宽度为 1 （像素）的线条  plt.plot(X, S, color="r", lw=4.0, linestyle="-")  plt.axis([-4,4,-1.2,1.2])  # 设置轴记号  plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],  [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])  plt.yticks([-1, 0, +1],  [r'$-1$', r'$0$', r'$+1$'])  # 在屏幕上显示  show() |

当我们设置记号的时候，我们可以同时设置记号的标签。注意这里使用了 LaTeX。



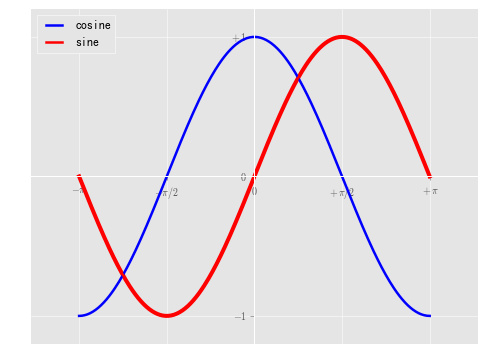
## 11、移动脊柱 坐标系

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ax = plt.gca() #gca表示获取当前坐标图对象的句柄，可以通过set(gca,...)设置当前坐标轴的外观，字体，颜色等  #plt.plot()实际上会通过plt.gca()获得当前的Axes对象ax，然后再调用ax.plot()方法实现真正的绘图。  # ax = Axes(0.125,0.1;0.775x0.8)  ax.spines['right'].set\_color('none')  ax.spines['top'].set\_color('none')  ax.xaxis.set\_ticks\_position('bottom')  ax.spines['bottom'].set\_position(('data',0))  ax.yaxis.set\_ticks\_position('left')  ax.spines['left'].set\_position(('data',0)) |

将坐标轴从边框部位转移至中间。

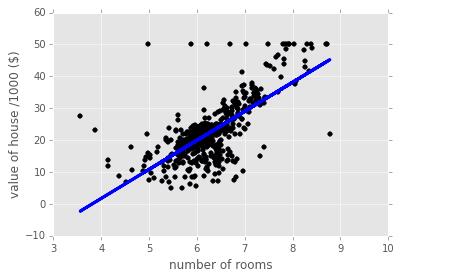
## 12、plt.legend()添加图例

|  |  |
| --- | --- |
| 1  2  3 | plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")  plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")  legend(loc='upper left') |



## 13、使用plt.style.use('ggplot')命令，可以作出ggplot风格的图片。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | # Import necessary packages  import pandas as pd  %matplotlib inline  import matplotlib.pyplot as plt  plt.style.use('ggplot')  from sklearn import datasets  from sklearn import linear\_model  import numpy as np  # Load data  boston = datasets.load\_boston()  yb = boston.target.reshape(-1, 1)  Xb = boston['data'][:,5].reshape(-1, 1)  # Plot data  plt.scatter(Xb,yb)  plt.ylabel('value of house /1000 ($)')  plt.xlabel('number of rooms')  # Create linear regression object  regr = linear\_model.LinearRegression()  # Train the model using the training sets  regr.fit( Xb, yb)  # Plot outputs  plt.scatter(Xb, yb, color='black')  plt.plot(Xb, regr.predict(Xb), color='blue',  linewidth=3)  plt.show() |



## 14、给特殊点做注释

好吧，又是注释，多个例子参考一下！

我们希望在 2π/32π/3 的位置给两条函数曲线加上一个注释。首先，我们在对应的函数图像位置上画一个点；然后，向横轴引一条垂线，以虚线标记；最后，写上标签。

# -\*- coding: UTF-8 -\*-

#import numpy as np

#import matplotlib.pyplot as plt

# 导入 matplotlib 的所有内容（nympy 可以用 np 这个名字来使用）

from pylab import \*

# 创建一个8 \* 6点（point）的图，并设置分辨率dpi为80

figure(figsize=(8,6), dpi=80)

# 创建一个新的 1 \* 1 的子图，接下来的图样绘制在其中的第 1 块（也是唯一的一块）

subplot(1,1,1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)

C,S = np.cos(X), np.sin(X)

# 绘制余弦曲线，使用蓝色的、连续的、宽度为 1 （像素）的线条

plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")

# 绘制正弦曲线，使用绿色的、连续的、宽度为 1 （像素）的线条

plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")

legend(loc='upper left')

#设置横纵轴的取值范围

plt.axis([-4,4,-1.2,1.2])

# 设置轴记号

xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],

[r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

yticks([-1, 0, +1],

[r'$-1$', r'$0$', r'$+1$'])

ax = gca() #gca表示获取当前坐标图对象的句柄，可以通过set(gca,...)设置当前坐标轴的外观，字体，颜色等

# ax = Axes(0.125,0.1;0.775x0.8) 不懂什么意思

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.xaxis.set\_ticks\_position('bottom')

ax.spines['bottom'].set\_position(('data',0))

ax.yaxis.set\_ticks\_position('left')

ax.spines['left'].set\_position(('data',0))

t = 2\*np.pi/3

# 作一条垂直于x轴的线段，由数学知识可知，横坐标一致的两个点就在垂直于坐标轴的直线上了。这两个点是起始点。

plot([t,t],[0,np.cos(t)], color ='blue', linewidth=2.5, linestyle="--") #第一个数组表示x的变化，第二个数组表示y的变化

scatter([t,],[np.cos(t),], 50, color ='blue')

annotate(r'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',

xy=(t, np.sin(t)), xycoords='data',

xytext=(+10, +30), textcoords='offset points', fontsize=16,

arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

plot([t,t],[0,np.sin(t)], color ='red', linewidth=2.5, linestyle="--")

scatter([t,],[np.sin(t),], 50, color ='red')

annotate(r'$\cos(\frac{2\pi}{3})=-\frac{1}{2}$',

xy=(t, np.cos(t)), xycoords='data',

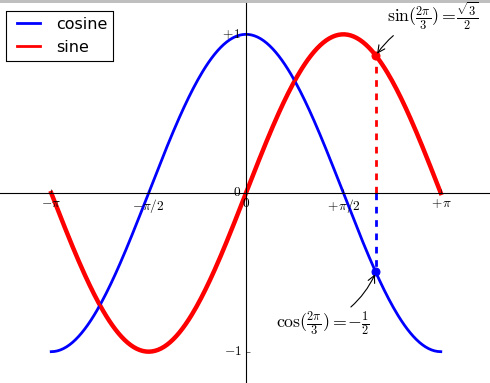
xytext=(-90, -50), textcoords='offset points', fontsize=16,

arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

# 在屏幕上显示

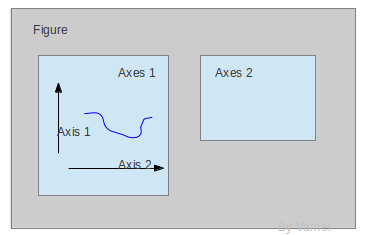
show()

代码在“第三章.plot”



## 15、plt.axes()

我们先来看什么是Figure和Axes对象。在matplotlib中，整个图像为一个Figure对象。在Figure对象中可以包含一个，或者多个Axes对象。每个Axes对象都是一个拥有自己坐标系统的绘图区域。其逻辑关系如下：



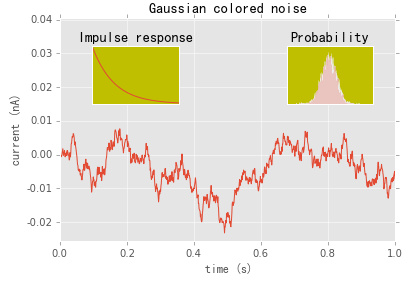
apes: 设置坐标轴边界和表面的颜色、坐标刻度值大小和网格的显示

* axes() by itself creates a default full subplot(111) window axis.
* axes(rect, axisbg=’w’) --where rect = [left, bottom, width, height] in normalized (0, 1) units. axisbg is the background color for the axis, default white.
* axes(h) where h is an axes instance makes h the current axis. An Axes instance is returned.

rect=[左, 下, 宽, 高] 规定的矩形区域,rect矩形简写，这里的数值都是以figure大小为比例，因此，若是要两个axes并排显示，那么axes[2]的左=axes[1].左+axes[1].宽，这样axes[2]才不会和axes[1]重叠。

show code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | http://matplotlib.org/examples/pylab\_examples/axes\_demo.html  import matplotlib.pyplot as plt  import numpy as np  # create some data to use for the plot  dt = 0.001  t = np.arange(0.0, 10.0, dt)  r = np.exp(-t[:1000]/0.05) # impulse response  x = np.random.randn(len(t))  s = np.convolve(x, r)[:len(x)]\*dt # colored noise  # the main axes is subplot(111) by default  plt.plot(t, s)  plt.axis([0, 1, 1.1\*np.amin(s), 2\*np.amax(s)])  plt.xlabel('time (s)')  plt.ylabel('current (nA)')  plt.title('Gaussian colored noise')  # this is an inset axes over the main axes  a = plt.axes([.65, .6, .2, .2], axisbg='y')  n, bins, patches = plt.hist(s, 400, normed=1)  plt.title('Probability')  plt.xticks([])  plt.yticks([])  # this is another inset axes over the main axes  a = plt.axes([0.2, 0.6, .2, .2], axisbg='y')  plt.plot(t[:len(r)], r)  plt.title('Impulse response')  plt.xlim(0, 0.2)  plt.xticks([])  plt.yticks([])  plt.show() |



## 16、plt.axis('equal')避免比例压缩为椭圆

（下文参考文献：<http://blog.csdn.net/u011367448/article/details/17370393）>

1、通过pyplot画散点图，调用scatter函数，同时该函数也可以画不同大小点的散点图。例子如下：

# -\*- coding : UTF-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

x = np.arange(0,20,2)

y = np.linspace(0,20,10) #在float中产生数组，10是数组元素的个数

plt.figure() #绘制图形窗口的大小

plt.scatter(x,y,c='r',marker='\*') #若要去除黑色轮廓，则可以使用edgecolor

plt.xlabel('X')

plt.ylabel('Y')

plt.title('$X\*Y$') #$--$的作用是产生斜体字

plt.show()

2．画曲线图，通过plot函数画曲线图，如下：

import numpy as np

import random

from matplotlib import pylab as pl

from matplotlib import pyplot as plt

#画一条曲线图

plt.figure(figsize=(8,6))   #设置图形的尺寸，单位为英尺

x = np.random.randn(10)\*10 #从正态分布中产生10个数，并将数值扩大十倍

y = np.pi\*np.sin(x) + 8

x1 = np.arange(0,10,1)

#lines = plt.plot(x,y)

plt.plot(x,y,label = '$np.pi\*np.sin(x)+8$',color='red',linewidth = 2)

plt.plot(x,y,'bo')

plt.title('Y=a\*sin(x)+b')

plt.ylabel('Function-Y')

plt.xlabel('Var-X')

plt.legend()   #图形的右上角显示图形的标签，和上面的plt.plo(x,y,label='')有关

#plt.setp(lines,color = 'r',linewidth = 2.5)

plt.show()

3、将图像画在多个轴上面，并在一个figure中。可以通过subplot，还有ubplot2grid等等方式均可实现。

import numpy as np

import matplotlib.pylab as plt'''将图形画在一个矩阵制定的表中，利用subplot，plot()中放有多个自变量和函数,画多坐标图'''def f(t):

    return np.exp(-t) \* np.cos(2\*np.pi\*t)

t1 = np.arange(0.0, 5.0, 0.1)

t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)

plt.subplot(211)  #subplot(numRows, numCols, plotNum),如：subplot(221)

plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.title('pylab first example',fontsize = 16)

plt.subplot(212)

plt.plot(t2, np.cos(2\*np.pi\*t2), 'r--')

plt.title('pylab second example',fontsize = 16)

plt.show()

4.使用text将文本表现在图中，并对图像之间进行设置，坐标设置等。

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

#通过对象的形式文本作图

fig = plt.figure()

fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')#figure的标题

ax = fig.add\_subplot(111)   #figure下的坐标图axfig.subplots\_adjust(top=0.85)

ax.set\_title('axes title')  #给ax设置标题

ax.set\_xlabel('xlabel')     #给ax设置横坐标

ax.set\_ylabel('ylabel')     #给ax设置纵坐标

#3,8表示写文字的起始位置，bbox是针对框图设置的，alpha表示方框颜色的深度，pad表示方框

ax.text(3, 8, 'boxed italics text in data coords', style='italic',

        bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

ax.text(2, 6, r'an equation: $E=mc^2$', fontsize=15)

ax.text(3, 2, unicode('unicode: Institut f\374r Festk\366rperphysik', 'latin-1'))

ax.text(0.95, 0.01, 'colored text in axes coords',

        verticalalignment='bottom', horizontalalignment='right',

        transform=ax.transAxes,

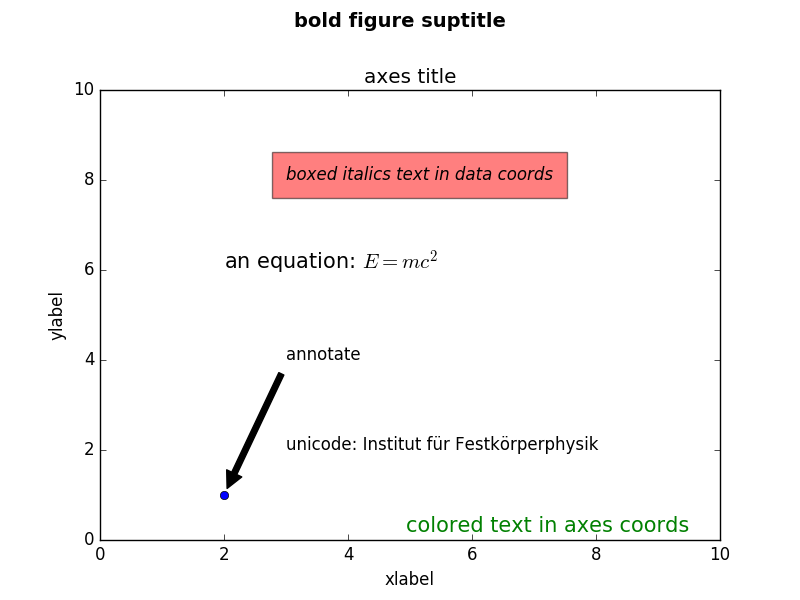
        color='green', fontsize=15)

ax.plot([2], [1], 'o')      #ax画图，以及下面添加注释ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),

arrowprops=dict(facecolor='black', shrink=0.05))

ax.axis([0, 10, 0, 10]) #设置横-纵轴的刻度

plt.show()



5、通过hist画直方图

import numpy as np

import matplotlib.pyplot as pltx = np.arange(0,1000,0.5)y = np.random.rand(2000)\*10

n,bins,patches = plt.hist(y,50,normed=1,alpha=0.8)

plt.title('Hist of Y')

plt.xlabel('Smarts')

plt.ylabel('NormProbability')

plt.text(2,0.12,'$\mu=10,\\sigma=20$')

plt.grid(True)

plt.show()

6、通过bar画条形图

import numpy as np

import matplotlib.pyplot as plt

x = np.arange(0,1000,0.5)

y = np.random.rand(2000)\*10n,bins,patches = plt.hist(y,50,normed=1,alpha=0.8)plt.title('Hist of Y')

plt.xlabel('Smarts')

plt.ylabel('NormProbability')

plt.text(2,0.12,'$\mu=10,\\sigma=20$')

plt.grid(True)

plt.show()

7、对图形中的重要点进行注释，采用annotate实现改功能。具体用法可以采用官方文档。如下：

# .\*. coding : utf-8 .\*.

import numpy as np

import matplotlib.pyplot as pltx = np.arange(-10,10,1)y = 2\*(x\*\*2)-5\*x+np.pi

maxy = min(y)

maxyx = x[np.argmin(y)]

plt.figure()

plt.plot(x,y,'-',color='red',linewidth=2.5,label='$2\*(x^2)-5\*x+np.pi$')

plt.annotate('Max y at points',xy=(maxyx,maxy),xytext=(maxyx+1,maxy+10),

             arrowprops=dict(facecolor='black', shrink=0.01),fontsize=12)

#xy为注释的点，xytext注释点文本的位置

plt.xlabel('X')

plt.ylabel('Y')

plt.xlim(-11,11)

plt.title('plot of y=x^2')

plt.show()

matplotlib官方文档中包含可很多画图的方法，如时间序列方面的处理，如：acorr；图像的处理，信号处理，数学公式等等。官方文（<http://matplotlib.org/py-modindex.html>）

## [17、plt.tick\_params()](https://blog.csdn.net/helunqu2017/article/details/78736554)

### 1.tick\_params语法 设置刻度标记大小

Axes.tick\_params(axis='both', \*\*kwargs)

参数:

axis : {‘x’, ‘y’, ‘both’} Axis on which to operate; default is ‘both’.

reset : bool If True, set all parameters to defaults before processing other keyword arguments. Default is False.

which : {‘major’, ‘minor’, ‘both’} Default is ‘major’; apply arguments to which ticks.

direction : {‘in’, ‘out’, ‘inout’} Puts ticks inside the axes, outside the axes, or both.

length : float Tick length in points.

width : float Tick width in points.

color : color Tick color; accepts any mpl color spec.

pad : float Distance in points between tick and label.

labelsize : float or str Tick label font size in points or as a string (e.g., ‘large’).

labelcolor : color Tick label color; mpl color spec.

colors : color Changes the tick color and the label color to the same value: mpl color spec.

zorder : float Tick and label zorder.

bottom, top, left, right : bool or {‘on’, ‘off’} controls whether to draw the respective ticks.

labelbottom, labeltop, labelleft, labelright : bool or {‘on’, ‘off’} controls whether to draw the respective tick labels.

labelrotation : float Tick label rotation

### 2.tick\_params例子:

参数axis的值为'x'、'y'、'both'，分别代表设置X轴、Y轴以及同时设置，默认值为'both'。

ax1.tick\_params(axis='x',width=2,colors='gold')

ax2.tick\_params(axis='y',width=2,colors='gold')

ax3.tick\_params(axis='both',width=2,colors='gold')

参数which的值为 'major'、'minor'、'both'，分别代表设置主刻度线、副刻度线以及同时设置，默认值为'major'

ax1.tick\_params(which='major',width=2,colors='gold')

ax2.tick\_params(which='minor',width=2,colors='gold')

ax3.tick\_params(which='both',width=2,colors='gold')

参数direction的值为'in'、'out'、'inout'，分别代表刻度线显示在绘图区内侧、外侧以及同时显示

ax1.tick\_params(direction='in',width=2,length=4,colors='gold')

ax2.tick\_params(direction='out',width=2,length=4,colors='gold')

ax3.tick\_params(direction='inout',width=2,length=4,colors='gold')

length和width

参数length和width分别用于设置刻度线的长度和宽度

ax2.tick\_params(width=4,colors='gold')

ax3.tick\_params(length=10,colors='gold')

参数pad用于设置刻度线与标签间的距离

ax2.tick\_params(pad=1,colors='gold')

ax3.tick\_params(pad=10,colors='gold')

参数color、labelcolor、colors分别用于设置刻度线的颜色、刻度线标签的颜色以及同时设置刻度线及标签颜色

ax1.tick\_params(width=4,color='gold')

ax2.tick\_params(width=4,labelcolor='gold')

ax3.tick\_params(width=4,colors='gold')

参数labelsize用于设置刻度线标签的字体大小

ax1.tick\_params(labelsize='medium')

ax2.tick\_params(labelsize='large')

ax3.tick\_params(labelsize=15)

参数bottom, top, left, right的值为布尔值，分别代表设置绘图区四个边框线上的的刻度线是否显示

ax1.tick\_params(bottom=False,top=True,width=4,colors='gold')

ax2.tick\_params(left=False,right=True,width=4,colors='gold')

ax3.tick\_params(top=True,right=True,width=4,colors='gold')

参数labelbottom, labeltop, labelleft, labelright的值为布尔值，分别代表设置绘图区四个边框线上的刻度线标签是否显示

ax1.tick\_params(labelbottom=False,labeltop=True,width=4,colors='gold')

ax2.tick\_params(labelleft=False,labelright=True,width=4,colors='gold')

ax3.tick\_params(labeltop=True,labelright=True,width=4,colors='gold')