

官方示范： 可以在这里查询图例和代码：<http://matplotlib.sourceforge.net/gallery.html#>

Python图表绘制：matplotlib绘图库入门

matplotlib 是 python 最著名的绘图库，它提供了一整套和 matlab 相似的命令 API，十分适合交互式地行制图。而且也可以方便地将它作为绘图控件，嵌入 GUI 应用程序中。

它的文档相当完备，并且 Gallery 页面中有上百幅缩略图，打开之后都有源程序。因此如果你需要绘制某种类型的图，只需要在这个页面中浏览/复制/粘贴一下，基本上都能搞定。

在 Linux 下比较著名的数据图工具还有 gnuplot，这个是免费的，Python 有一个包可以调用 gnuplot，但是语法比较不习惯，而且画图质量不高。

而 Matplotlib 则比较强：Matlab 的语法、python 语言、latex 的画图质量（还可以使用内嵌的 latex 引擎绘制的数学公式）。

Matplotlib.pyplot 快速绘图

快速绘图和面向对象方式绘图

matplotlib 实际上是一套面向对象的绘图库，它所绘制的图表中的每个绘图元素，例如线条 Line2D、文字 Text、刻度等在内存中都有一个对象与之对应。

为了方便快速绘图 matplotlib 通过 pyplot 模块提供了一套和 MATLAB 类似的绘图 API，将众多绘图对象所构成的复杂结构隐藏在这套 API 内部。我们只需要调用 pyplot 模块所提供的函数就可以实现快速绘图以及设置图表的各种细节。pyplot 模块虽然用法简单，但不适合在较大的应用程序中使用。

为了将面向对象的绘图库包装成只使用函数的调用接口，pyplot 模块的内部保存了当前图表以及当前子图等信息。当前的图表和子图可以使用 plt.gcf() 和 plt.gca() 获得，分别表示 "Get Current Figure" 和 "Get Current Axes"。在 pyplot 模块中，许多函数都是对当前的 Figure 或 Axes 对象进行处理，比如说：

```
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。

可以在 Ipython 中输入类似 "plt.plot??" 的命令查看 pyplot 模块的函数是如何对各种绘图对象进行包装的。

配置属性

matplotlib 所绘制的图表的每个组成部分都和一个对象对应，我们可以通过调用这些对象的属性设置方法 set_*() 或者 pyplot 模块的属性设置函数 setp() 设置它们的属性值。

因为 matplotlib 实际上是一套面向对象的绘图库，因此也可以直接获取对象的属性

配置文件

绘制一幅图需要对许多对象的属性进行配置，例如颜色、字体、线型等等。我们在绘图时，并没有逐一对这些属性进行配置，许多都直接采用了 matplotlib 的缺省配置。

matplotlib 将这些缺省配置保存在一个名为“matplotlibrc”的配置文件中，通过修改配置文件，我们可以修改图表的缺省样式。配置文件的读入可以使用 `rc_params()`，它返回一个配置字典；在 matplotlib 模块载入时会调用 `rc_params()`，并把得到的配置字典保存到 `rcParams` 变量中；matplotlib 将使用 `rcParams` 字典中的配置进行绘图；用户可以直接修改此字典中的配置，所做的改变会反映到此后创建的绘图元素。

绘制多子图（快速绘图）

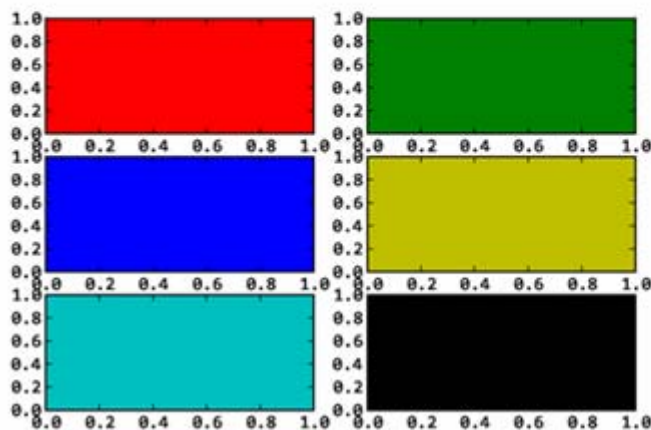
Matplotlib 里的常用类的包含关系为 `Figure -> Axes -> (Line2D, Text, etc.)` 一个 `Figure` 对象可以包含多个子图(`Axes`)，在 matplotlib 中用 `Axes` 对象表示一个绘图区域，可以理解为子图。

可以使用 `subplot()` 快速绘制包含多个子图的图表，它的调用形式如下：

```
subplot(numRows, numCols, plotNum)
```

`subplot` 将整个绘图区域等分为 `numRows` 行 * `numCols` 列个子区域，然后按照从左到右，从上到下的顺序对每个子区域进行编号，左上的子区域的编号为 1。如果 `numRows`，`numCols` 和 `plotNum` 这三个数都小于 10 的话，可以把它们缩写为一个整数，例如 `subplot(323)` 和 `subplot(3,2,3)` 是相同的。`subplot` 在 `plotNum` 指定的区域中创建一个轴对象。如果新创建的轴和之前创建的轴重叠的话，之前的轴将被删除。

```
for idx, color in enumerate("rgbyck"):
    plt.subplot(321+idx, axisbg=color)
plt.show()
```



`subplot()` 返回它所创建的 `Axes` 对象，我们可以将它用变量保存起来，然后用 `sca()` 交替让它们成为当前 `Axes` 对象，并调用 `plot()` 在其中绘图。

绘制多图表（快速绘图）

如果需要同时绘制多幅图表，可以给 `figure()` 传递一个整数参数指定 `Figure` 对象的序号，如果序号所指定的 `Figure` 对象已经存在，将不创建新的对象，而只是让它成为当前的 `Figure` 对象。

```

import numpy as np
import matplotlib.pyplot as plt

plt.figure(1) # 创建图表 1

plt.figure(2) # 创建图表 2

ax1 = plt.subplot(211) # 在图表 2 中创建子图 1

ax2 = plt.subplot(212) # 在图表 2 中创建子图 2

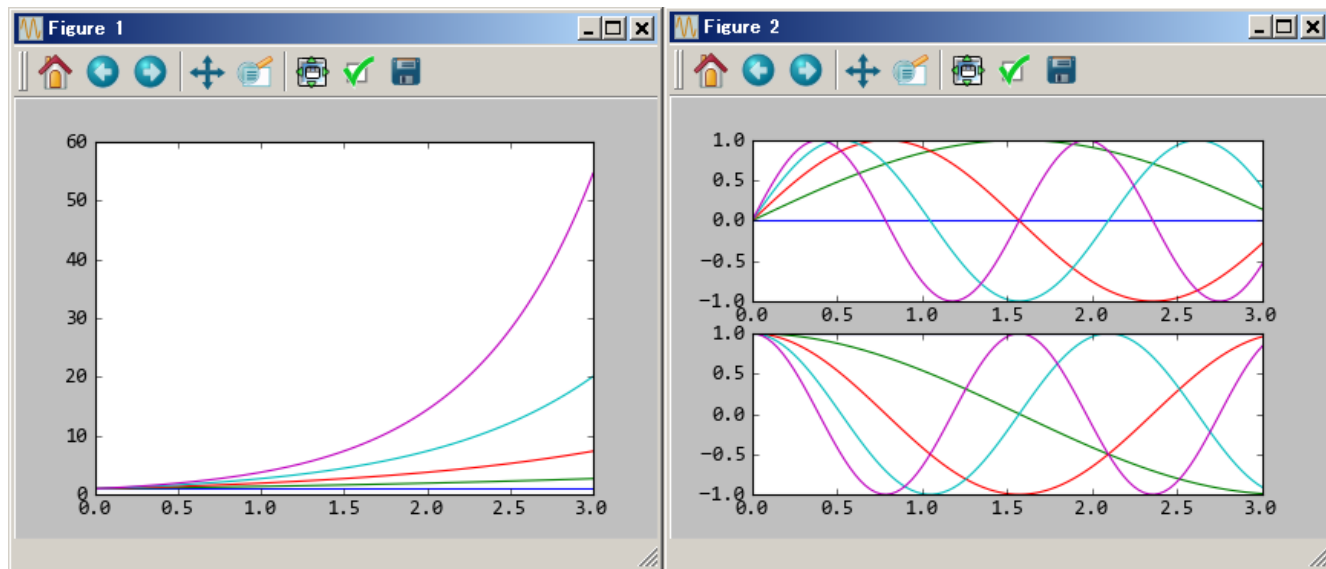
x = np.linspace(0, 3, 100)
for i in xrange(5):

    plt.figure(1) #❶ # 选择图表 1
    plt.plot(x, np.exp(i*x/3))

    plt.sca(ax1) #❷ # 选择图表 2 的子图 1
    plt.plot(x, np.sin(i*x))

    plt.sca(ax2) # 选择图表 2 的子图 2
    plt.plot(x, np.cos(i*x))
plt.show()

```



在图表中显示中文

matplotlib 的缺省配置文件中所使用的字体无法正确显示中文。为了让图表能正确显示中文，可以有几种解决方案。

1. 在程序中直接指定字体。
2. 在程序开头修改配置字典 rcParams。
3. 修改配置文件。

面向对象画图

matplotlib API 包含有三层，Artist 层处理所有的高层结构，例如处理图表、文字和曲线等的绘制和布局。通常我们只和 Artist 打交道，而不需要关心底层的绘制细节。

直接使用 Artists 创建图表的标准流程如下：

- 创建 Figure 对象
- 用 Figure 对象创建一个或者多个 Axes 或者 Subplot 对象
- 调用 Axes 等对象的方法创建各种简单类型的 Artists

```
import matplotlib.pyplot as plt
X1 = range(0, 50)
Y1 = [num**2 for num in X1] # y = x^2
X2 = [0, 1]
Y2 = [0, 1] # y = x
Fig = plt.figure(figsize=(8,4)) # Create a 'figure' instance
Ax = Fig.add_subplot(111) # Create a 'axes' instance in the figure
Ax.plot(X1, Y1, X2, Y2) # Create a Line2D instance in the axes
Fig.show()
Fig.savefig("test.pdf")
```

参考：

《Python科学计算》([Numpy视频](#)) [matplotlib-绘制精美的图表\(快速绘图\)\(面向对象绘图\)](#)
(深入浅出适合系统学习)

什么是 [Matplotlib](#) （主要讲面向对象绘图，对于新手可能有点乱）

Matplotlib.pylab 快速绘图

matplotlib 还提供了一个名为 pylab 的模块，其中包括了许多 NumPy 和 pyplot 模块中常用的函数，方便用户快速进行计算和绘图，十分适合在 IPython 交互式环境中使用。这里使用下面的方式载入 pylab 模块：

```
>>> import pylab as pl
```

1 安装 numpy 和 matplotlib

```
>>> import numpy
>>> numpy.__version__

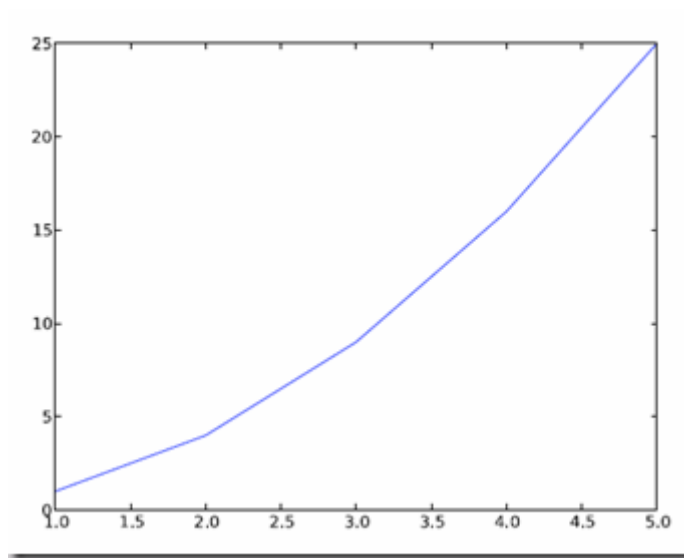
>>> import matplotlib
>>> matplotlib.__version__
```

2 两种常用图类型: Line and scatter plots(使用 plot() 命令), histogram(使用 hist() 命令)

2.1 折线图&散点图 Line and scatter plots

2.1.1 折线图 Line plots(关联一组 x 和 y 值的直线)

```
import numpy as np
import pylab as pl
x = [1, 2, 3, 4, 5]# Make an array of x values
y = [1, 4, 9, 16, 25]# Make an array of y values for each x value
pl.plot(x, y)# use pylab to plot x and y
pl.show()# show the plot on the screen
```



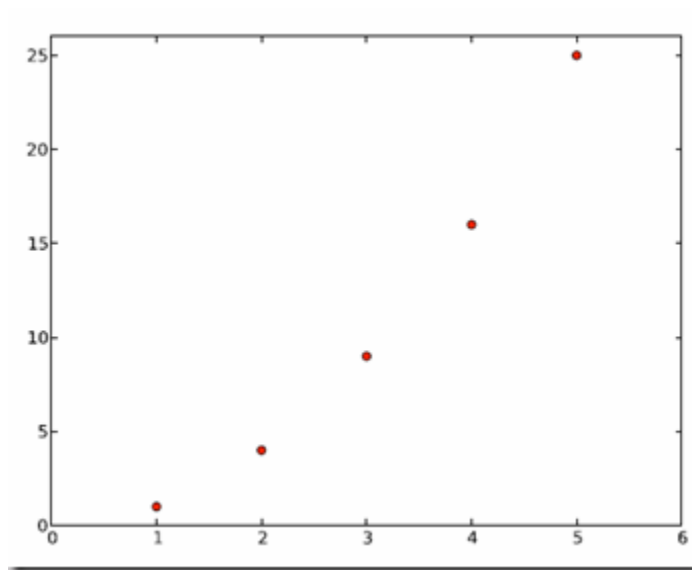
2.1.2 散点图 Scatter plots

把 pl.plot(x, y) 改成 pl.plot(x, y, 'o') 即可, 下图的蓝色版本

2.2 美化 Making things look pretty

2.2.1 线条颜色 Changing the line color

红色: 把 pl.plot(x, y, 'o') 改成 pl.plot(x, y, 'or')



character	color
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

2.2.2 线条样式 Changing the line style

虚线: `plot(x,y, '--')`

linestyle	description
'-'	solid
'--'	dashed
'-.'	dash_dot
':'	dotted
'None'	draw nothing
' '	draw nothing
' '	draw nothing

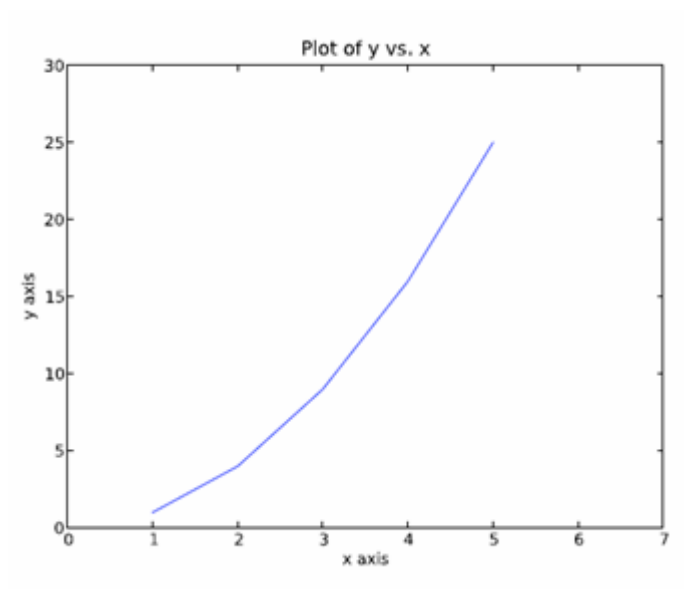
2.2.3 marker 样式 Changing the marker style

蓝色星型 markers: `plot(x,y, 'b*')`

's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin diamond marker

2.2.4 图和轴标题以及轴坐标限度 Plot and axis titles and limits

```
import numpy as np
import pylab as pl
x = [1, 2, 3, 4, 5]# Make an array of x values
y = [1, 4, 9, 16, 25]# Make an array of y values for each x value
pl.plot(x, y)# use pylab to plot x and y
pl.title('Plot of y vs. x')# give plot a title
pl.xlabel('x axis')# make axis labels
pl.ylabel('y axis')
pl.xlim(0.0, 7.0)# set axis limits
pl.ylim(0.0, 30.)
pl.show()# show the plot on the screen
```



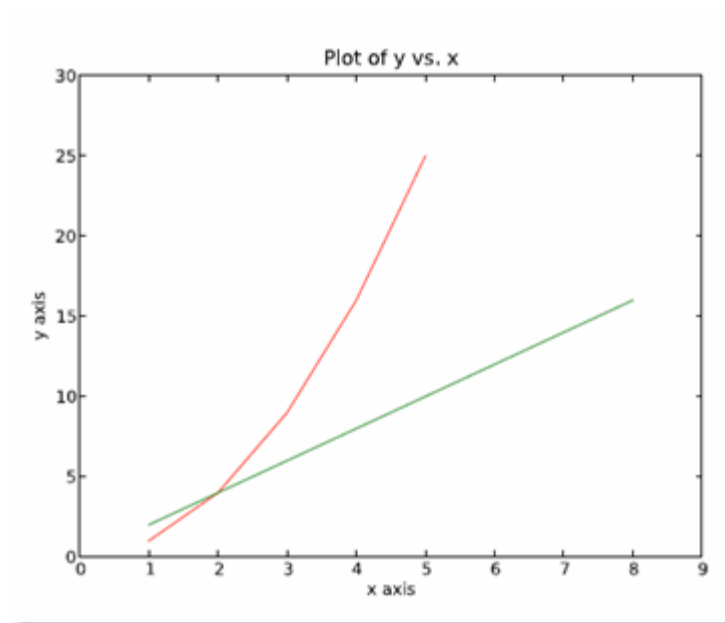
2.2.5 在一个坐标系上绘制多个图 Plotting more than one plot on the same set of axes

做法是很直接的，依次作图即可：

```

import numpy as np
import pylab as pl
x1 = [1, 2, 3, 4, 5]# Make x, y arrays for each graph
y1 = [1, 4, 9, 16, 25]
x2 = [1, 2, 4, 6, 8]
y2 = [2, 4, 8, 12, 16]
pl.plot(x1, y1, 'r')# use pylab to plot x and y
pl.plot(x2, y2, 'g')
pl.title('Plot of y vs. x')# give plot a title
pl.xlabel('x axis')# make axis labels
pl.ylabel('y axis')
pl.xlim(0.0, 9.0)# set axis limits
pl.ylim(0.0, 30.)
pl.show()# show the plot on the screen

```



2.2.6 图例 Figure legends

```
pl.legend((plot1, plot2), ('label1, label2'), 'best', numpoints=1)
```

其中第三个参数表示图例放置的位置: 'best' 'upper right', 'upper left', 'center', 'lower left', 'lower right'.

如果在当前 figure 里 plot 的时候已经指定了 label, 如 `plt.plot(x,z,label="$\cos(x^2)$")`, 直接调用 `plt.legend()` 就可以了哦。

```

import numpy as np
import pylab as pl
x1 = [1, 2, 3, 4, 5]# Make x, y arrays for each graph
y1 = [1, 4, 9, 16, 25]
x2 = [1, 2, 4, 6, 8]

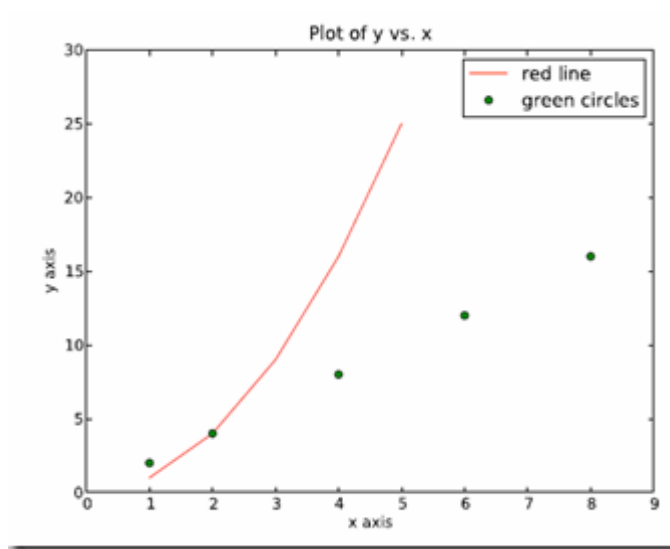
```



```

y2 = [2, 4, 8, 12, 16]
plot1 = pl.plot(x1, y1, 'r')# use pylab to plot x and y : Give your plots names
plot2 = pl.plot(x2, y2, 'go')
pl.title('Plot of y vs. x')# give plot a title
pl.xlabel('x axis')# make axis labels
pl.ylabel('y axis')
pl.xlim(0.0, 9.0)# set axis limits
pl.ylim(0.0, 30.)
pl.legend([plot1, plot2], ('red line', 'green circles'), 'best', numpoints=1)# make
legend
pl.show()# show the plot on the screen

```



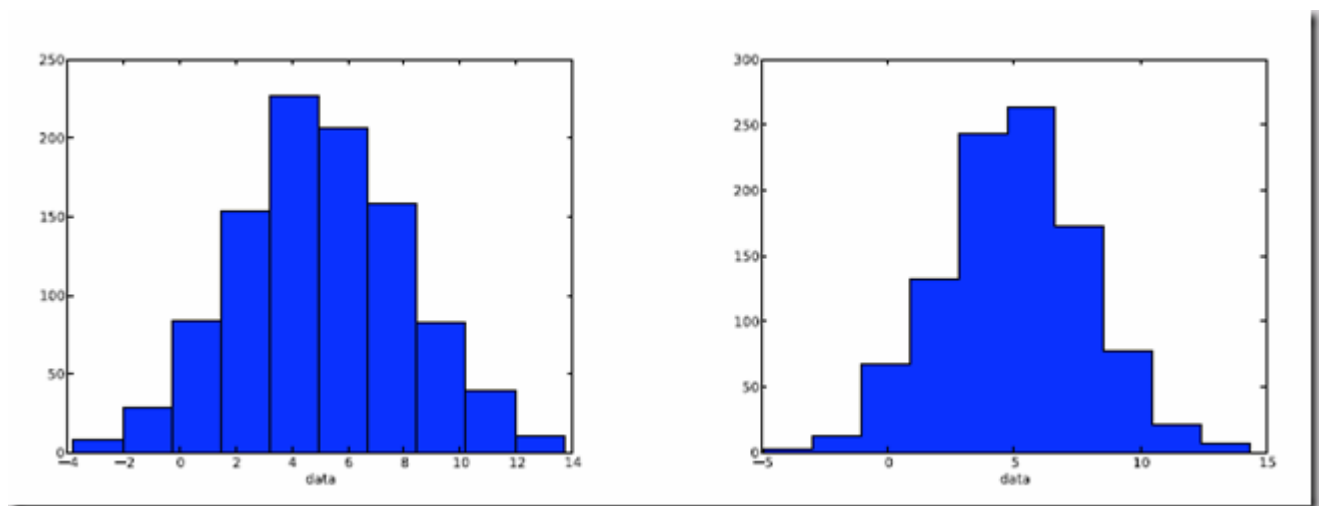
2.3 直方图 Histograms

```

import numpy as np
import pylab as pl
# make an array of random numbers with a gaussian distribution with
# mean = 5.0
# rms = 3.0
# number of points = 1000
data = np.random.normal(5.0, 3.0, 1000)
# make a histogram of the data array
pl.hist(data)
# make plot labels
pl.xlabel('data')
pl.show()

```

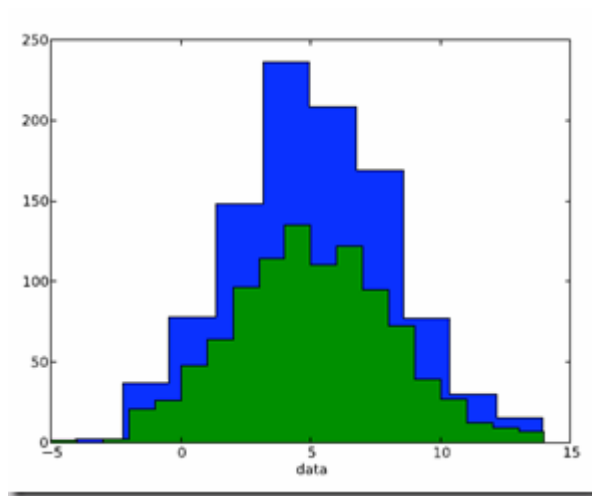
如果不要黑色轮廓可以改为 `pl.hist(data, histtype='stepfilled')`



2.3.1 自定义直方图 bin 宽度 Setting the width of the histogram bins manually

增加这两行

```
bins = np.arange(-5., 16., 1.) #浮点数版本的 range
pl.hist(data, bins, histtype='stepfilled')
```



3 同一画板上绘制多幅子图 Plotting more than one axis per canvas

如果需要同时绘制多幅图表的话，可以是给 figure 传递一个整数参数指定图表的序号，如果所指定

序号的绘图对象已经存在的话，将不创建新的对象，而只是让它成为当前绘图对象。

```
fig1 = pl.figure(1)
```

```
pl.subplot(211)
```

subplot(211)把绘图区域等分为 2 行*1 列共两个区域，然后在区域 1(上区域)中创建一个轴对

象. `pl.subplot(212)`在区域 2(下区域)创建一个轴对象。

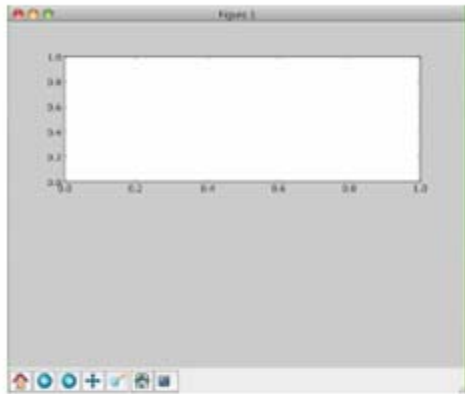


Figure 9: Showing subplot(211)

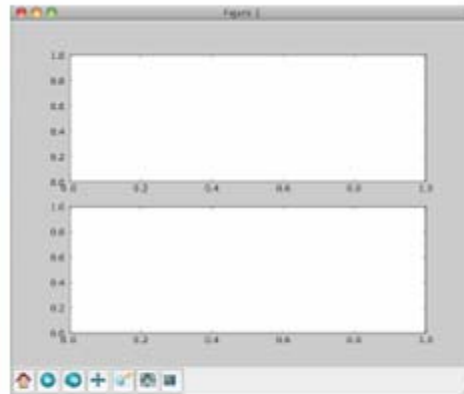


Figure 10: Showing subplot(212)

You can play around with plotting a variety of layouts. For example, Fig. 11 is created using the following commands:

```
f1 = pl.figure(1)
pl.subplot(221)
pl.subplot(222)
pl.subplot(212)
```

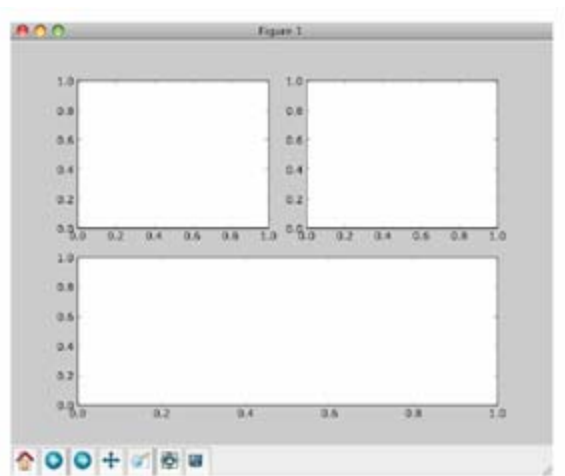


Figure 11: Playing with subplot

当绘图对象中有多个轴的时候，可以通过工具栏中的 **Configure Subplots** 按钮，交互式地调节轴之间的间距和轴与边框之间的距离。如果希望在程序中调节的话，可以调用 `subplots_adjust` 函数，它有 `left`, `right`, `bottom`, `top`, `wspace`, `hspace` 等几个关键字参数，这些参数的值都是 0 到 1 之间的小数，它们是以绘图区域的宽高为 1 进行规范化之后的坐标或者长度。

```
pl.subplots_adjust(left=0.08, right=0.95, wspace=0.25, hspace=0.45)
```

4 绘制文件中的数据 Plotting data contained in files

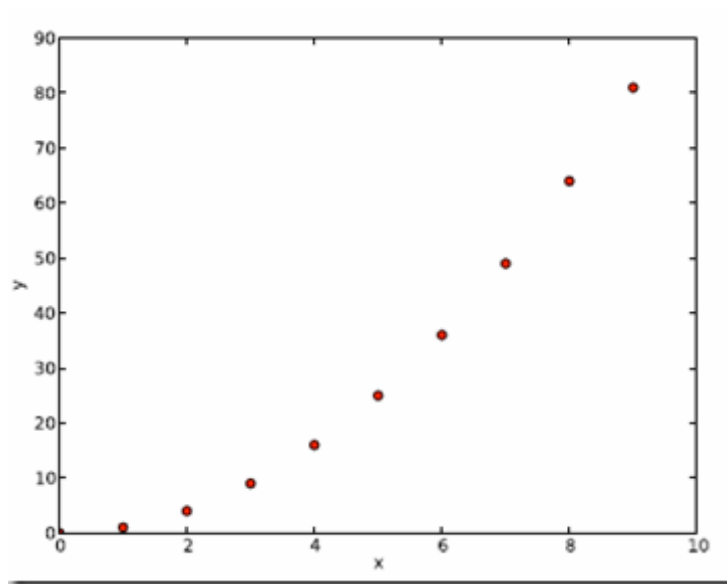
4.1 从 Ascii 文件中读取数据 Reading data from ascii files

读取文件的方法很多，这里只介绍一种简单的方法，更多的可以参考官方文档和[NumPy快速处理数据\(文件存取\)](#)。

numpy 的 loadtxt 方法可以直接读取如下文本数据到 numpy 二维数组

```
*****  
  
# fakedata.txt  
0 0  
1 1  
2 4  
3 9  
4 16  
5 25  
6 36  
7 49  
8 64  
9 81  
0 0  
1 1  
2 4  
3 9  
4 16  
5 25  
6 36  
7 49  
8 64  
9 81  
  
*****
```

```
import numpy as np  
import pylab as pl  
# Use numpy to load the data contained in the file  
# 'fakedata.txt' into a 2-D array called data  
data = np.loadtxt('fakedata.txt')  
# plot the first column as x, and second column as y  
pl.plot(data[:,0], data[:,1], 'ro')  
pl.xlabel('x')  
pl.ylabel('y')  
pl.xlim(0.0, 10.)  
pl.show()
```



4.2 写入数据到文件 Writing data to a text file

写文件的方法也很多，这里只介绍一种可用的写入文本文件的方法，更多的可以参考官方文档。

```
import numpy as np
# Let's make 2 arrays (x, y) which we will write to a file
# x is an array containing numbers 0 to 10, with intervals of 1
x = np.arange(0.0, 10., 1.)
# y is an array containing the values in x, squared
y = x*x
print 'x = ', x
print 'y = ', y
# Now open a file to write the data to
# 'w' means open for 'writing'
file = open('testdata.txt', 'w')
# loop over each line you want to write to file
for i in range(len(x)):
    # make a string for each line you want to write
    # '\t' means 'tab'
    # '\n' means 'newline'
    # 'str()' means you are converting the quantity in brackets to a string type
    txt = str(x[i]) + '\t' + str(y[i]) + '\n'
    # write the txt to the file
    file.write(txt)
# Close your file
file.close()
```

这部分是翻译自: [Python Plotting Beginners Guide](#)

对 LaTeX 数学公式的支持

Matplotlib 对 LaTeX 有一定的支持，如果记得使用 `raw` 字符串语法会很自然：

```
xlabel(r"$\frac{x^2}{y^4}$")
```

在 `matplotlib` 里面，可以使用 LaTeX 的命令来编辑公式，只需要在字符串前面加一个“`r`”即可

Here is a simple example:

```
# plain text  
plt.title('alpha > beta')
```

produces “alpha > beta”.

Whereas this:

```
# math text  
plt.title(r'$\alpha > \beta$')
```

produces “ $\alpha > \beta$ ”.

这里给大家看一个简单的例子。

```
import matplotlib.pyplot as plt  
  
x = arange(1,1000,1)  
r = -2  
c = 5  
y = [5*(a**r) for a in x]  
  
fig = plt.figure()  
  
ax = fig.add_subplot(111)  
ax.loglog(x,y,label = r"$y = \frac{1}{2\sigma_1^2},$  
c=5,\sigma_1=-2$")  
ax.legend()  
ax.set_xlabel(r"x")  
ax.set_ylabel(r"y")
```

程序执行结果如图 3 所示，这实际上是一个 `power-law` 的例子，有兴趣的朋友可以继续 `google` 之。

再看一个《用 Python 做科学计算》中的简单例子，下面的两行程序通过调用 `plot` 函数在当前的绘图对象中进行绘图：

```
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)
```

```
plt.plot(x,z,"b--",label="$cos(x^2)$")
```

`plot` 函数的调用方式很灵活，第一句将 `x,y` 数组传递给 `plot` 之后，用关键字参数指定各种属性：

- **label**：给所绘制的曲线一个名字，此名字在图示(legend)中显示。只要在字符串前后添加"\$"符号，`matplotlib` 就会使用其内嵌的 **latex 引擎** 绘制的数学公式。
- **color**：指定曲线的颜色
- **linewidth**：指定曲线的宽度

详细的可以参考 `matplotlib` 官方教程：

[Writing mathematical expressions](#)

- [Subscripts and superscripts](#)
- [Fractions, binomials and stacked numbers](#)
- [Radicals](#)
- [Fonts](#)
 - [Custom fonts](#)
- [Accents](#)
- [Symbols](#)
- [Example](#)

[Text rendering With LaTeX](#)

- [usetex with unicode](#)
- [Postscript options](#)
- [Possible hangups](#)
- [Troubleshooting](#)

有几个问题：

- `matplotlib.rcParams` 属性字典
- 想要它正常工作，在 `matplotlibrc` 配置文件中需要设置 `text.markup = "tex"`。
- 如果你希望图表中所有的文字（包括坐标轴刻度标记）都是 LaTeX'd, 需要在 `matplotlibrc` 中设置 `text.usetex = True`。如果你使用 LaTeX 撰写论文，那么这一点对于使图表和论文中其余部分保持一致是很有用的。
- 在 `matplotlib` 中使用中文字符串时记住要用 `unicode` 格式，例如：`u"测试中文显示"`

[matplotlib使用小结](#)

[LaTeX科技排版](#)

对数坐标轴

在实际中，我们可能经常会用到对数坐标轴，这时可以用下面的三个函数来实现

`ax.semilogx(x,y)` #x 轴为对数坐标轴

`ax.semilogy(x,y)` #y 轴为对数坐标轴

`ax.loglog(x,y)` #双对数坐标轴

学习资源

Gnuplot 的介绍

- [Gnuplot简介](#)
- IBM: [gnuplot](#) 让您的数据可视化,Linux 上的数据可视化工具
- [利用Python绘制论文图片: Gnuplot,pylab](#)

官方英文资料:

- [matplotlib下载及API手册地址](#)
- [Screenshots: example figures](#)
- [Gallery: Click on any image to see full size image and source code](#)
- [matplotlib所使用的数学库numpy下载及API手册](#)

IBM: [基于 Python Matplotlib 模块的高质量图形输出](#)(2005 年的文章有点旧)

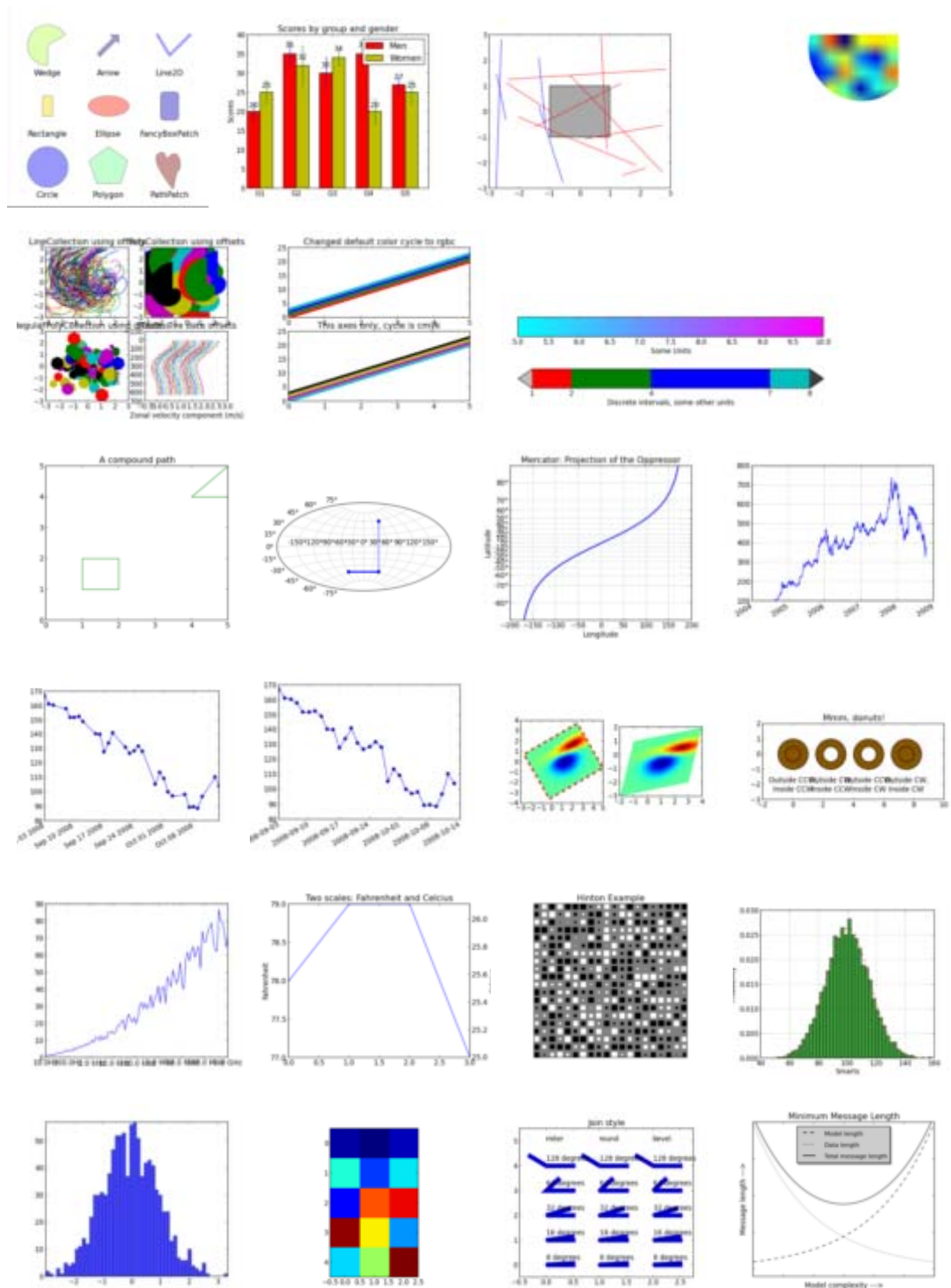
[matplotlib技巧集](#)(绘制不连续函数的不连续点; 参数曲线上绘制方向箭头; 修改缺省刻度数目; Y轴不同区间使用不同颜色填充的曲线区域。)

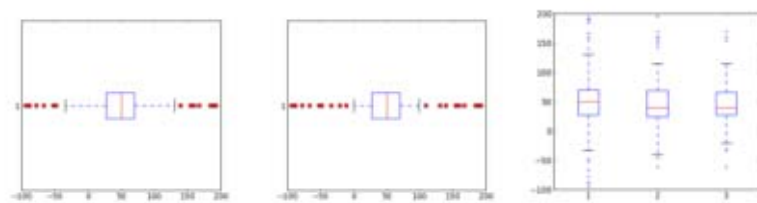
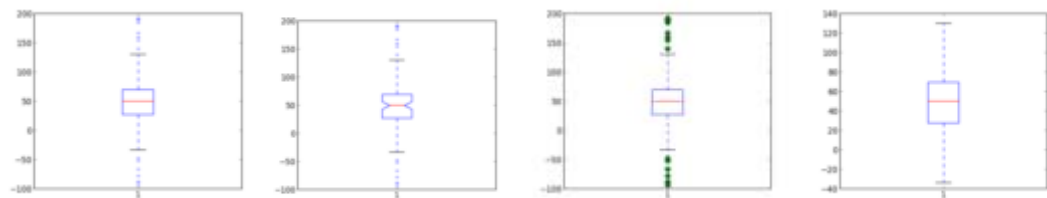
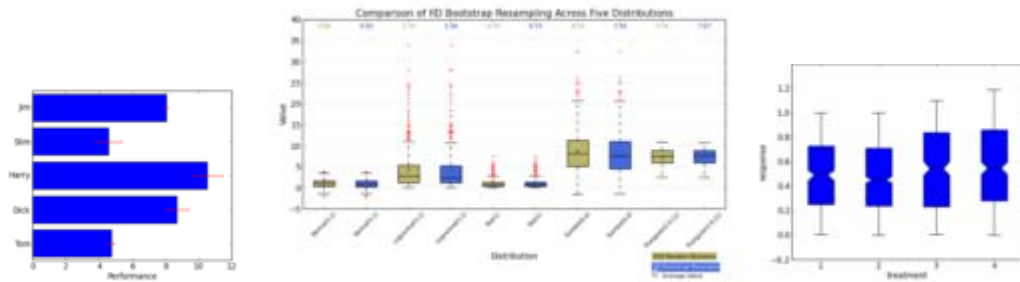
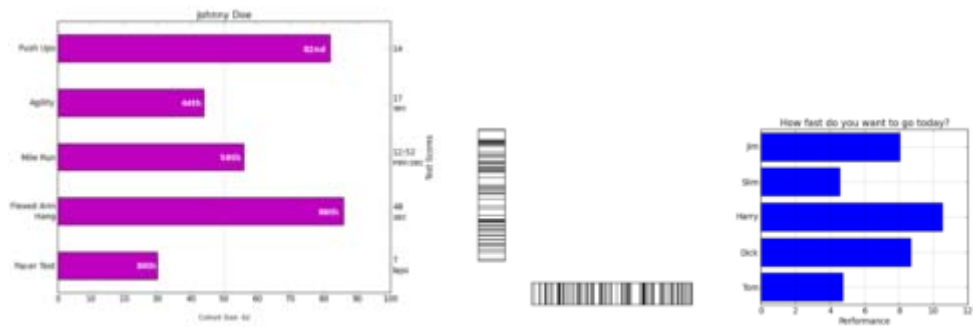
[Python: 使用matp绘制不连续函数的不连续点; 参数曲线上绘制方向箭头; 修改缺省刻度数目; Y轴不同区间使用不同颜色填充的曲线区域。](#)[lotlib绘制图表](#)

[matplotlib输出图象的中文显示问题](#)

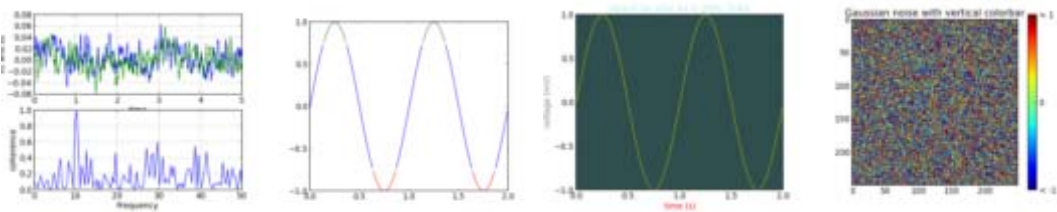
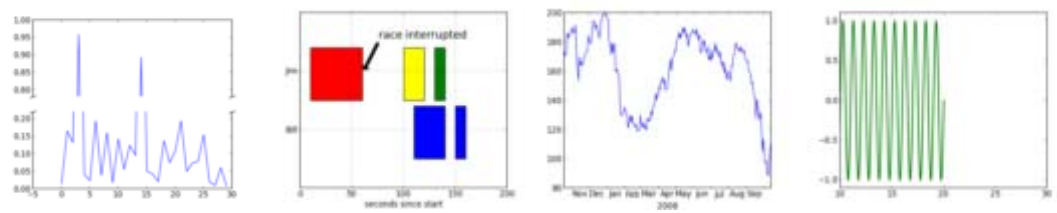
[matplotlib图表中图例大小及字体相关问题](#)

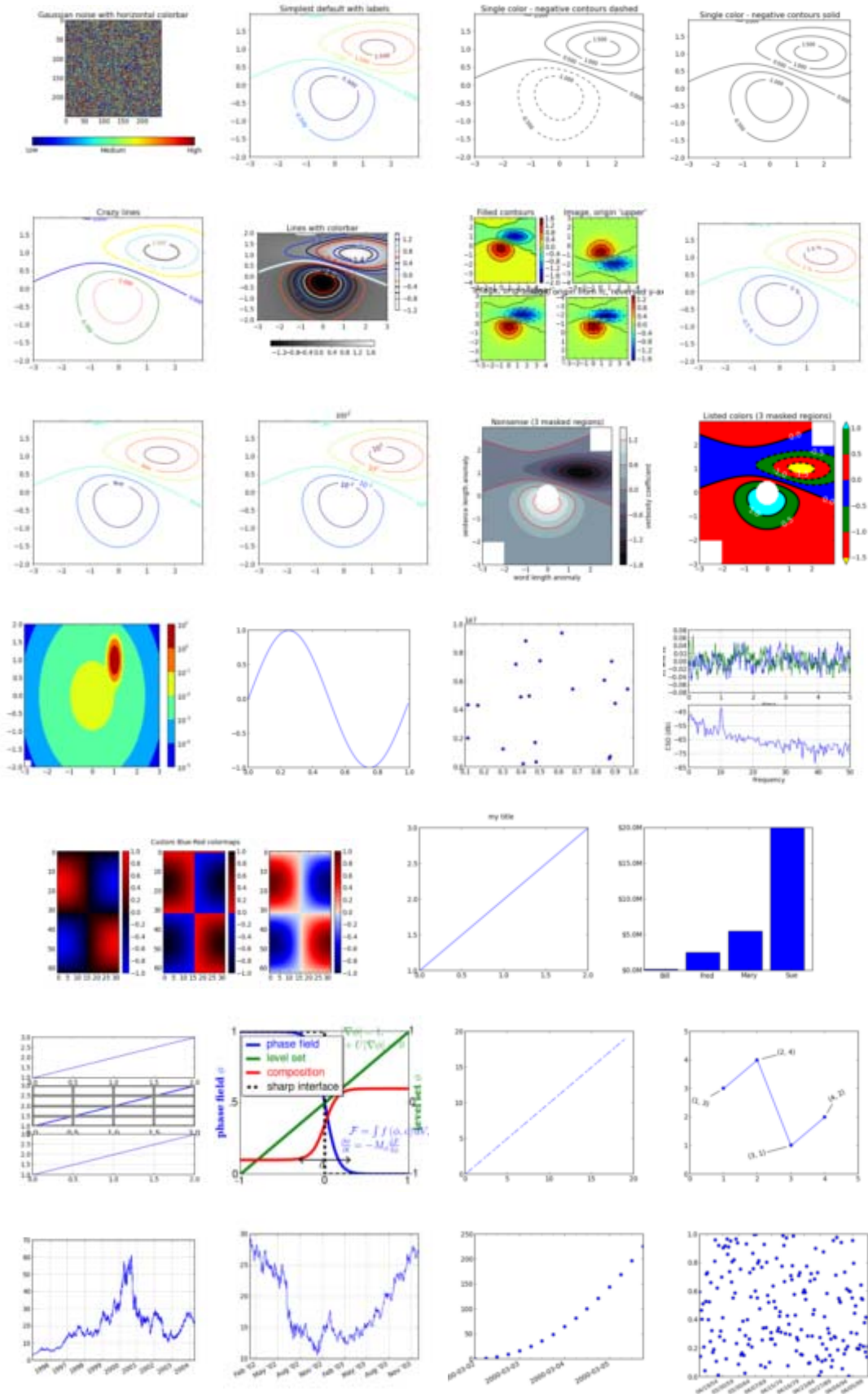
Click on any image to see full size image and source code

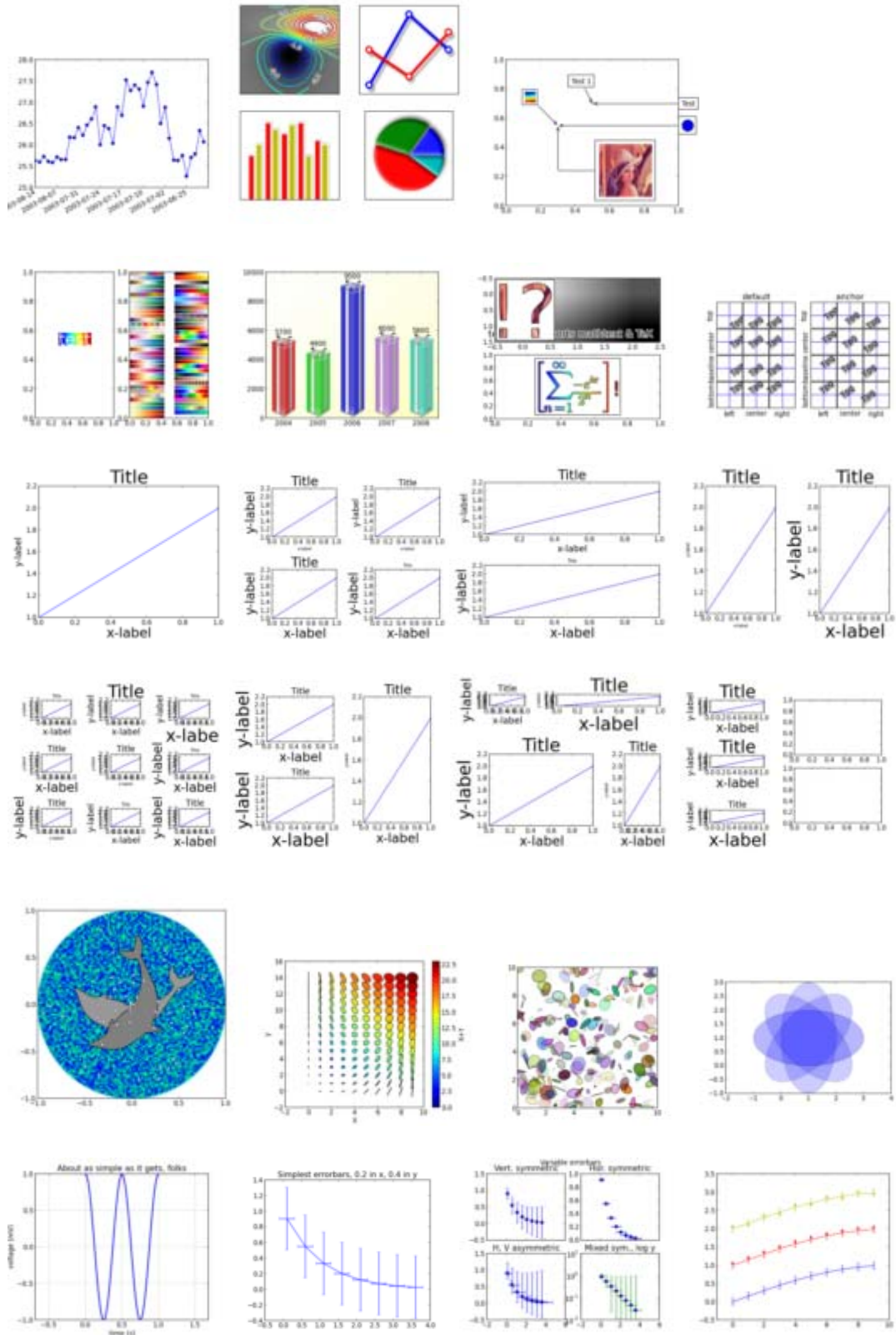


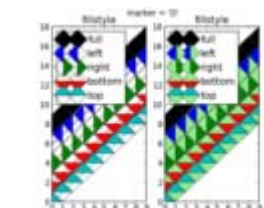
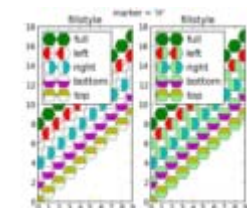
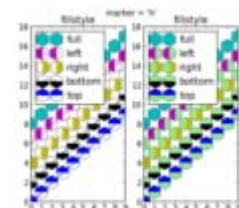
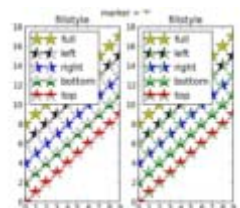
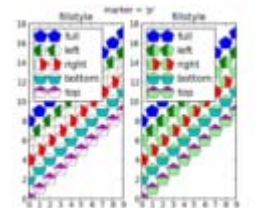
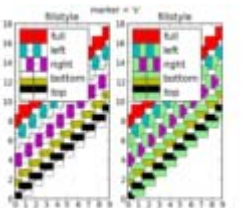
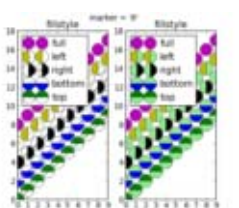
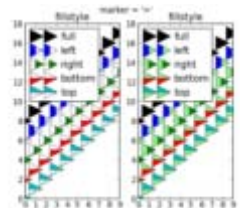
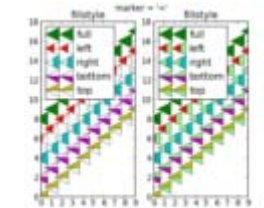
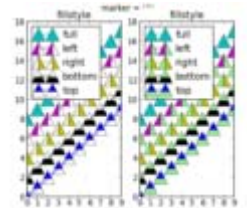
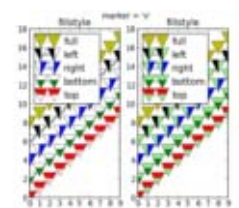
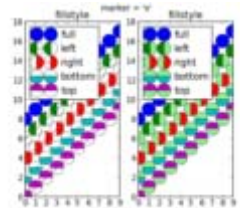
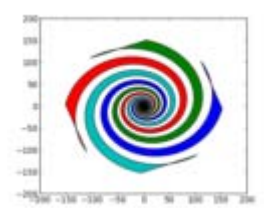
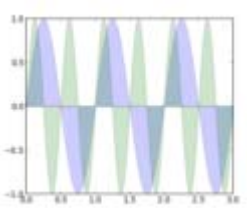
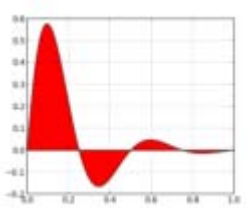
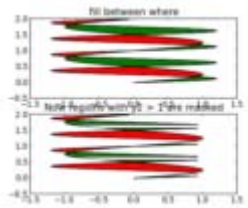
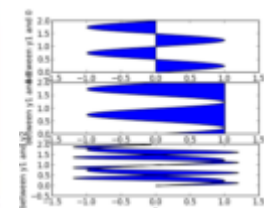
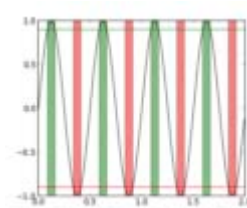
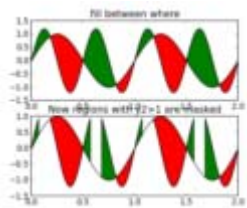
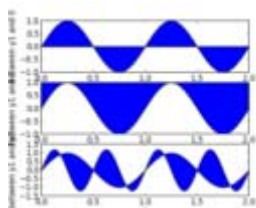
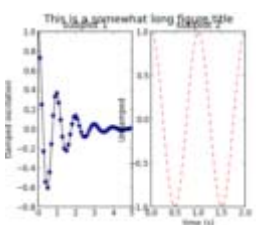
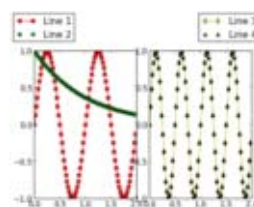
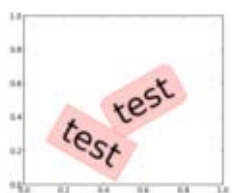
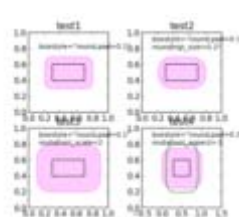
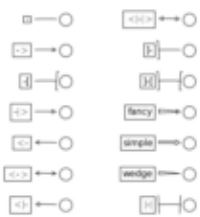
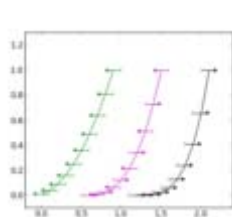


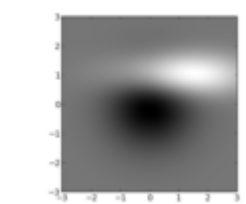
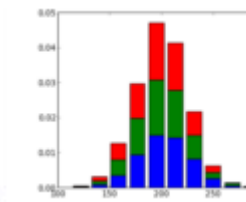
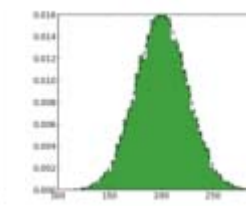
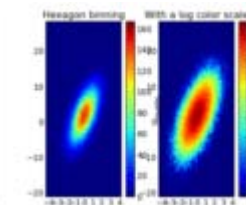
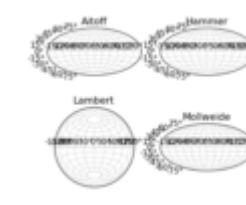
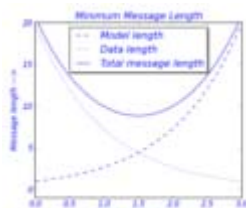
Distance Histograms by Category

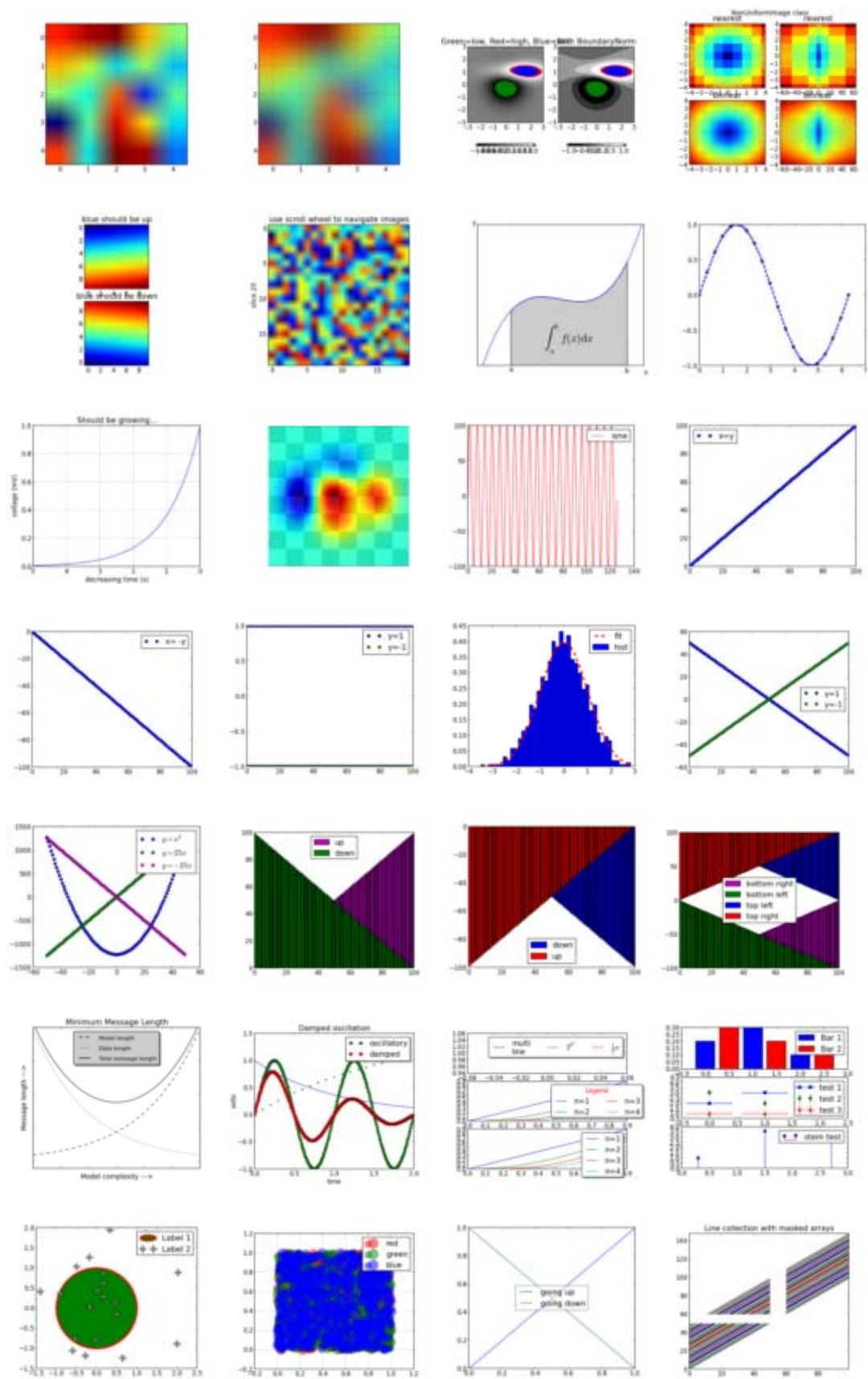


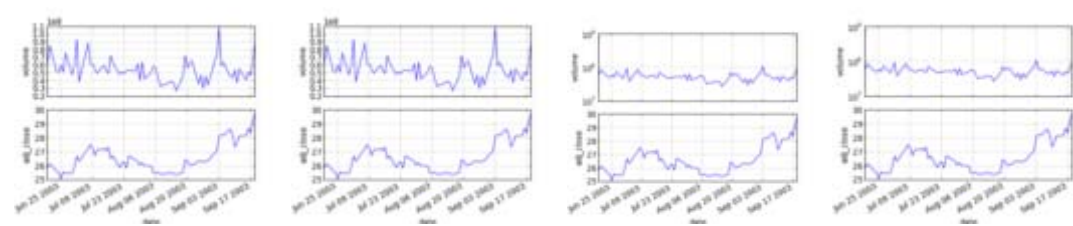
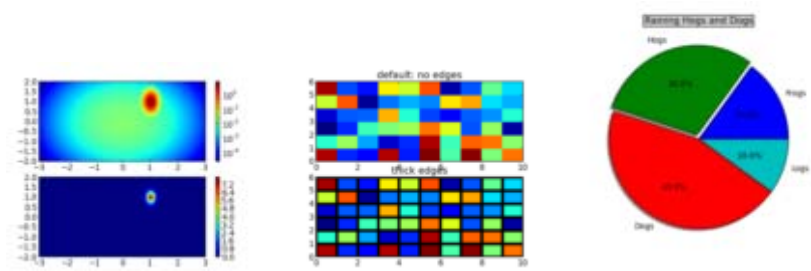
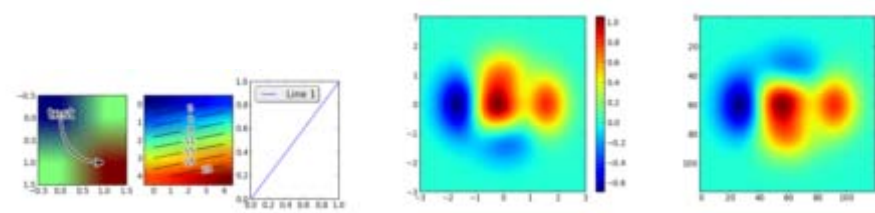
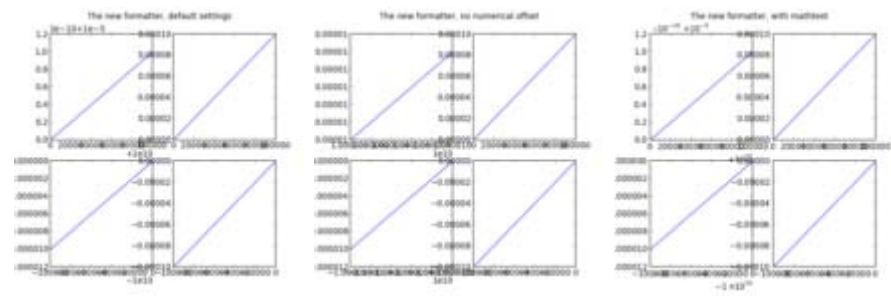
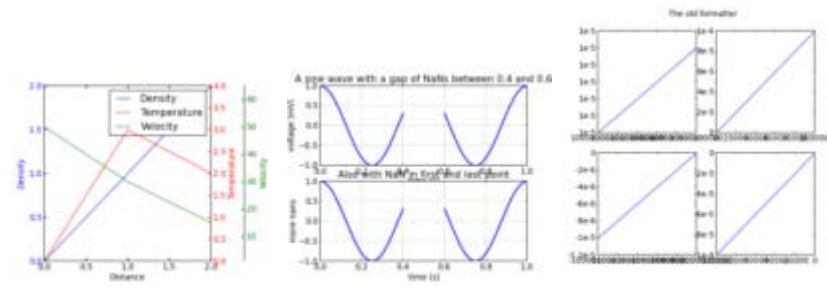
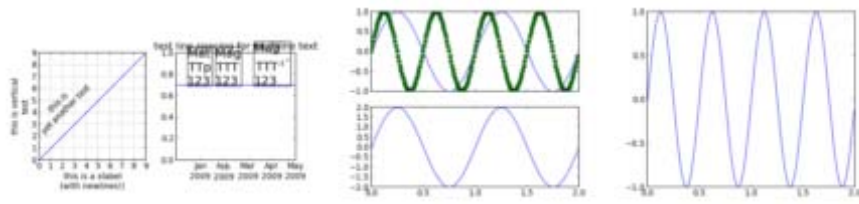


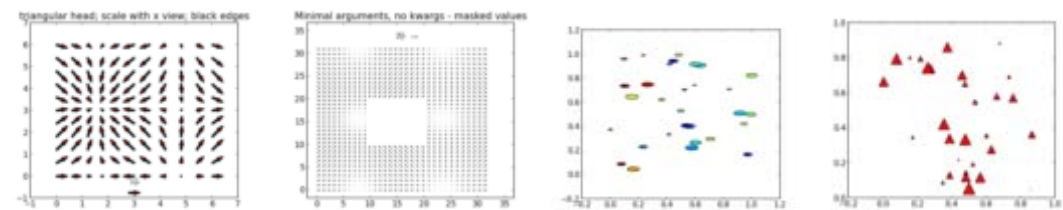
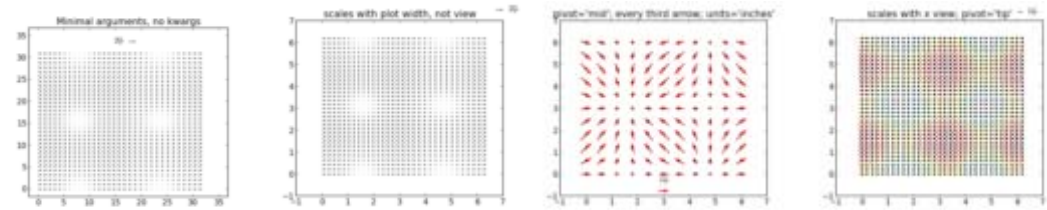
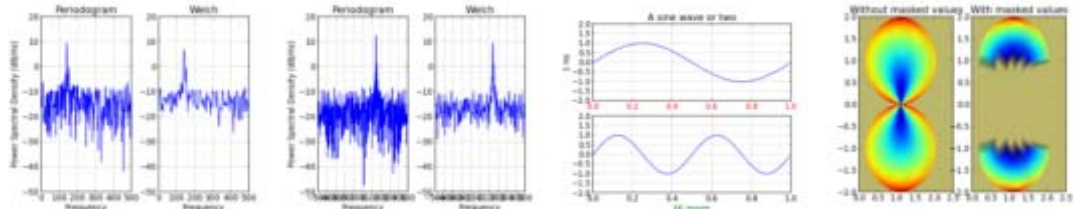
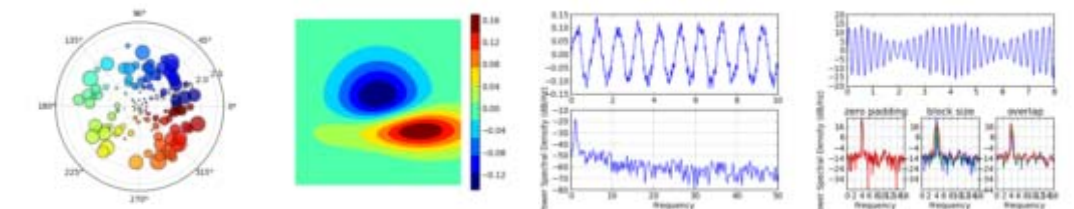
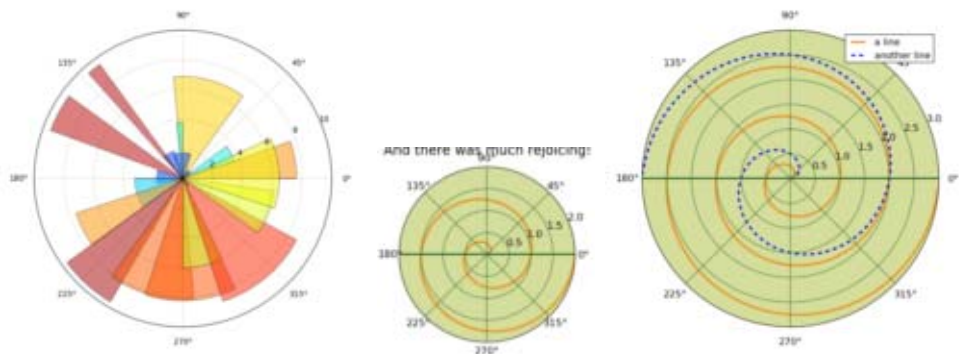
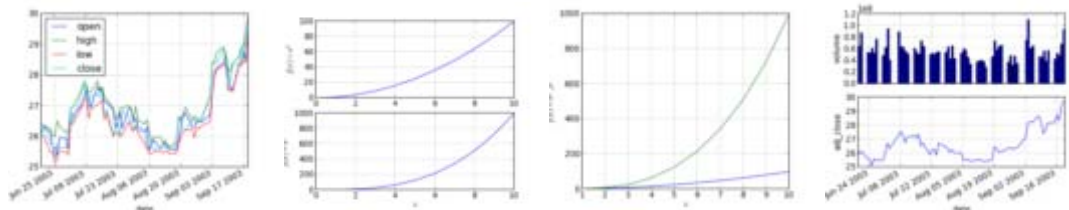


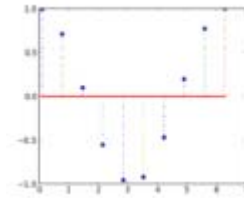
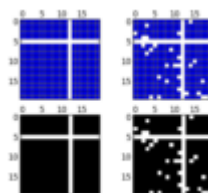
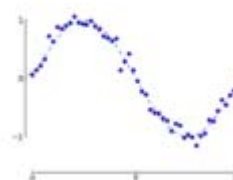
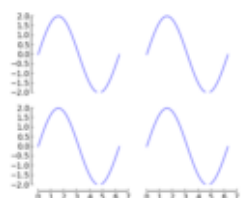
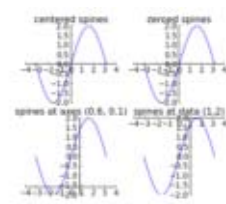
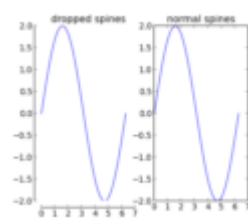
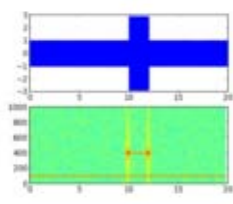
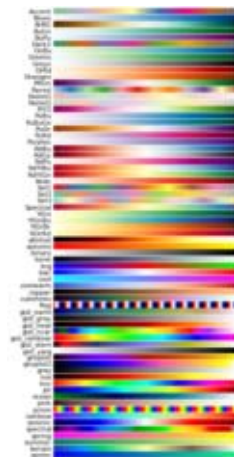
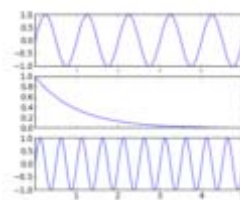
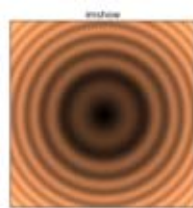
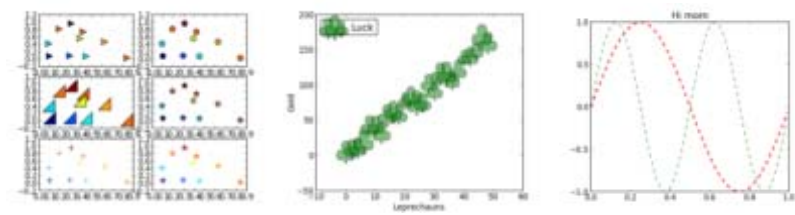
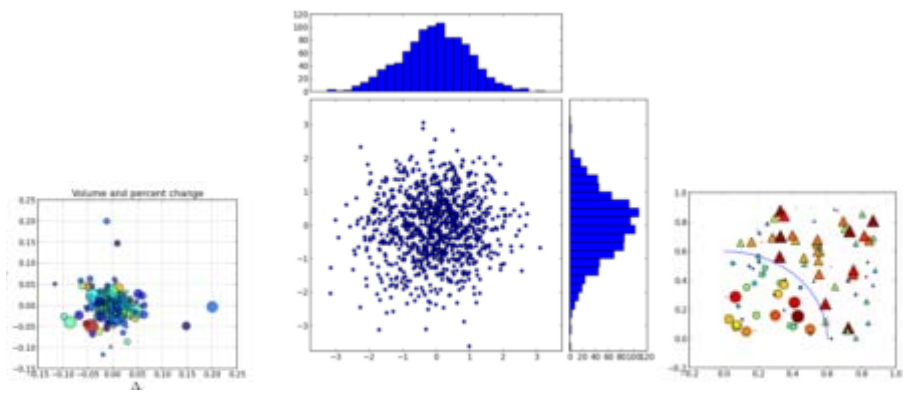


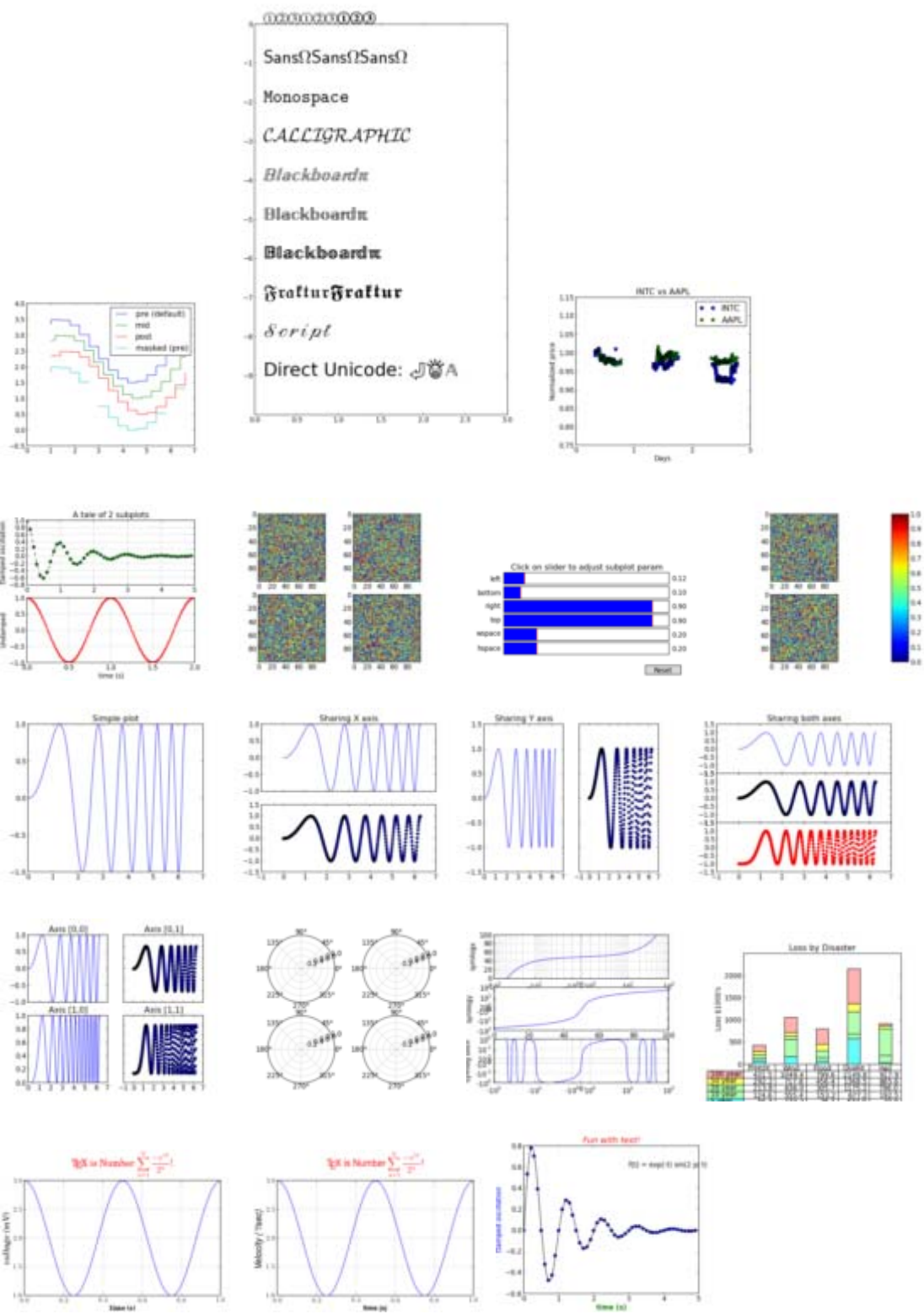


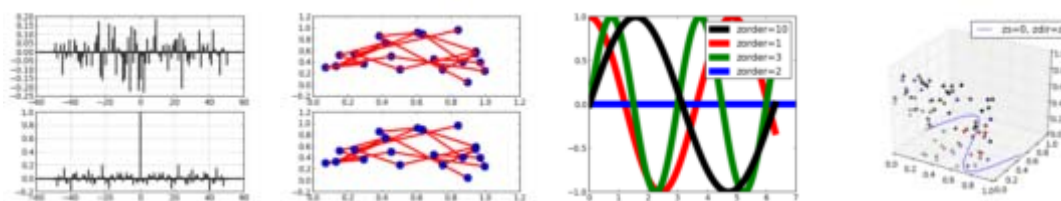
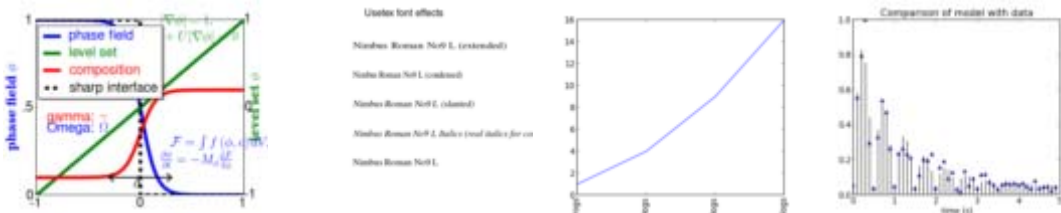
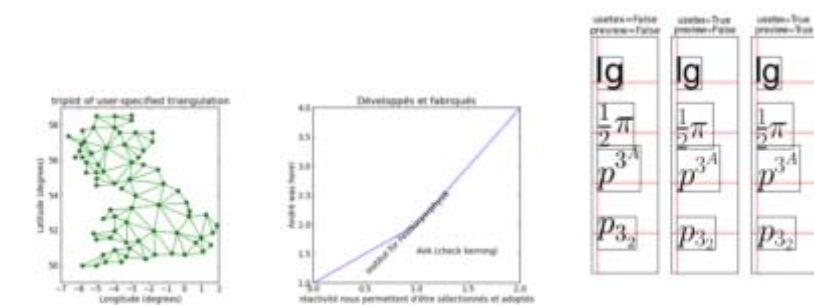
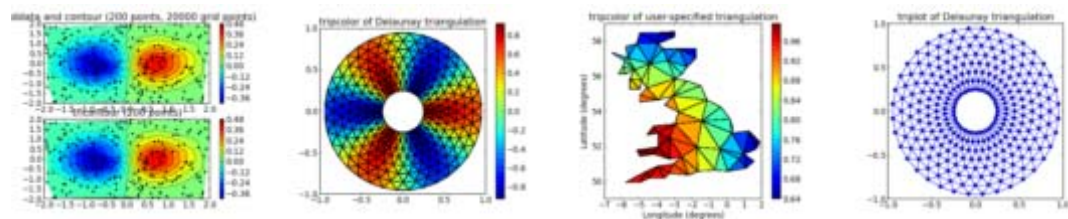
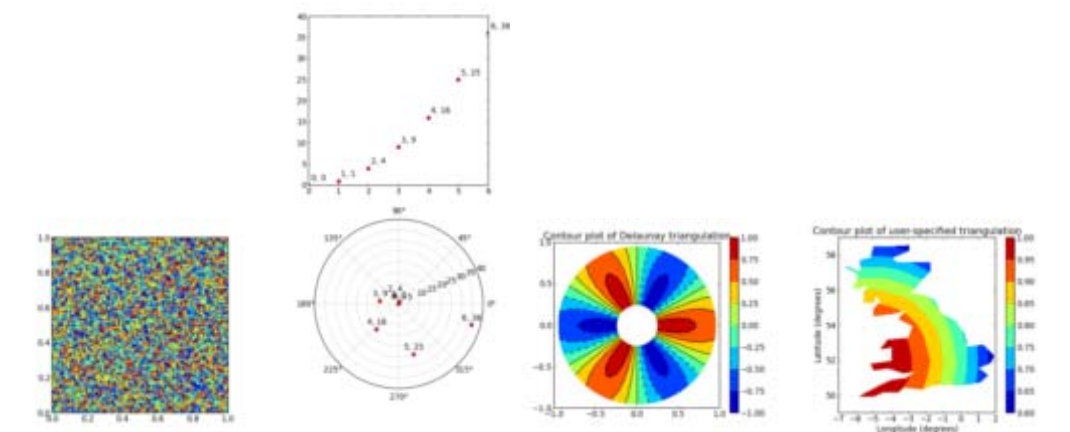
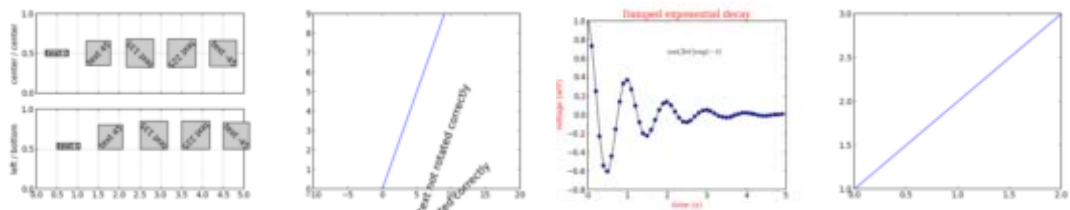


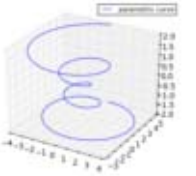
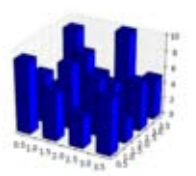
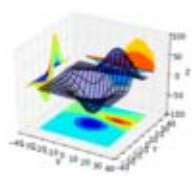
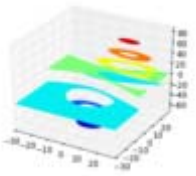
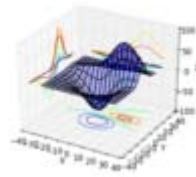
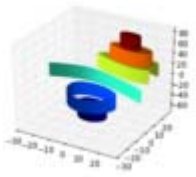
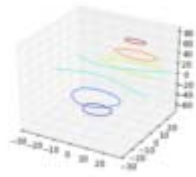
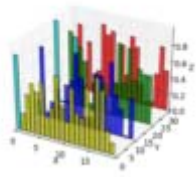












A table of 2 subplots

