Tutorial Zero: Data Visualization

实验前的准备

本次实验我们载入一些Python的安装包，如下：

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
```

背景

在本课程中，为说明了所构建模型的优良性或者展示模型的结果，我们常常需要使用图象来展示结果。使用图象的目的是直接或间接表达绘图者的观点，而非套用各种模版。

任务

在本次实验中，我们需要解决以下任务：

- 如何生成可以绘制图象的数据?
- 如何绘制简单图象?
- 如何生成具有个人风格的图象?
- 如何保存图象?

解决方案

Task1：如何生成可以绘制图象的数据?

在数据可视化中，我们往往需要生成数据。这是绘制图象的基础。

第一，我们可以利用numpy包中的linspace来生成固定区间上的等间隔数据。比较下面两段代码，我们可以了解linspace这个函数。

```
x1 = np.linspace(start = 0, stop = 1,num=11)
x1
```

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])
```

```
x2 = np.linspace(start = 0, stop = 1,num=10,endpoint = False)
x2
```

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

在numpy.linspace这个函数中，我们需要了解常用的参数：

1. start：区间左端点
2. stop：区间右端点
3. num：输出的数据个数（默认为50）
4. endpoint：是否为闭区间（默认为True）

还有其他参数，可以通过help(numpy.linspace)来自行挖掘。

第二，我们可以利用numpy包中的random.normal来生成标准正态分布的数据（伪随机数）。通过下面一段代码，我们可以了解random.normal这个函数。

```
x3 = np.random.normal(size = 10)
x3
```

```
array([ 0.92013307,  0.41624808,  0.68829865, -0.0327092 ,  0.54254451,
       -0.00588783, -1.36174511, -0.31745127, -2.32676449,  0.87955434])
```

除了标准正态分布之外，我们还可以生成其他连续分布的函数，如贝塔分布、卡方分布、指数分布、F分布、伽马分布等。

Task2：如何绘制简单图象？

在这个任务中，利用matplotlib.pyplot来绘制图象。这里我们介绍两种基础图象——折线图和散点图。

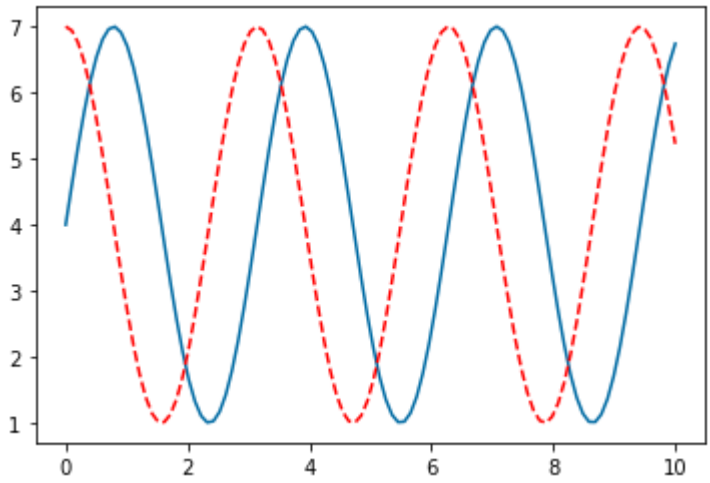
首先，我们介绍如何绘制折线图（line plot）。折线图是最基础的图象，其也可以衍生为时序图、概率密度函数图等。

以下我们在[0,10]区间上绘制两个函数f(x) = 4+3sin(2x)和g(x)=4+3cos(2x)。

```
x = np.linspace(0, 10, 100)
y1 = 4 + 3 * np.sin(2*x)
y2 = 4 + 3 * np.cos(2*x)

plt.style.use("tableau-colorblind10")
# 26 style
#'Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background',
#'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn',
#'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette',
'seaborn-darkgrid',
#'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-
pastel',
#'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-
```

```
whitegrid',
#'tableau-colorblind10'
fig, ax = plt.subplots()
ax.plot(x,y1)
ax.plot(x,y2,'r--') # r: red; --: dash line
plt.show()
```

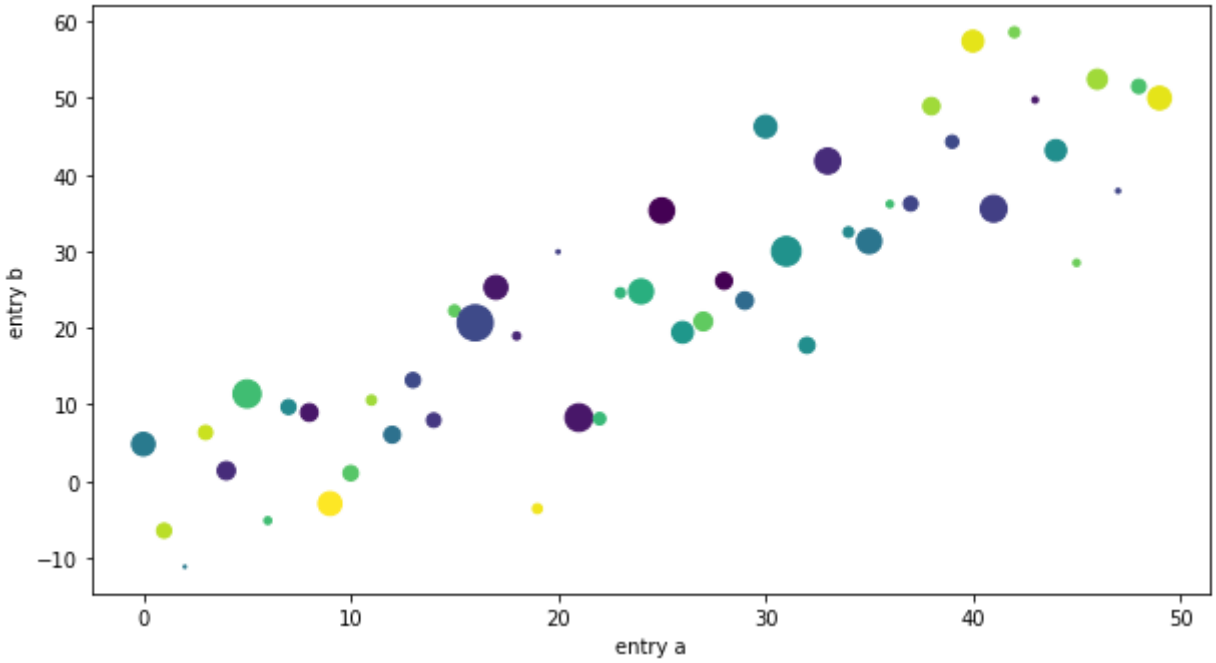


其次，我们介绍如何绘制散点图（scatter plot）。

```
np.random.seed(19680801)  # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100


fig, ax = plt.subplots(figsize=(10, 5.4))
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
```

```
Text(0, 0.5, 'entry b')
```



在ax.scatter这个函数中c表示颜色（color），s表示大小（size）。

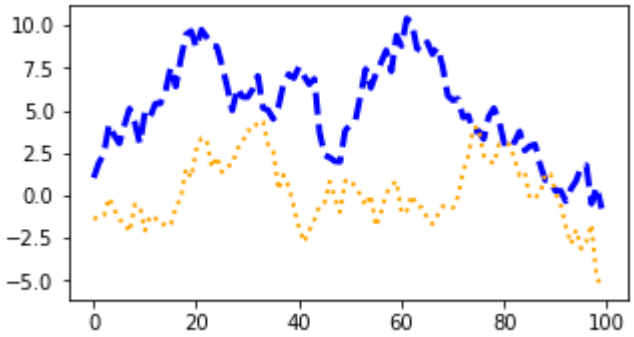
Task3：如何生成具有个人风格的图象？

这里我们主要从以下几个角度来介绍。

第一，线的风格。在set_linestyle这个参数中有以下几种常见的线的风格：

- '-' or 'solid'：实线；
- '--' or 'dashed'：虚线；
- '-.' or 'dashdot'：点划线；
- ':' or 'dotted'：点线；
- 'none', 'None', ' ', or ''：不绘制

```
# generate data
np.random.seed(19680801)
data1, data2, data3, data4 = np.random.randn(4, 100)
# line style
fig, ax = plt.subplots(figsize=(5, 2.7))
x = np.arange(len(data1))
ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')
l, = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)
l.set_linestyle(':')
```

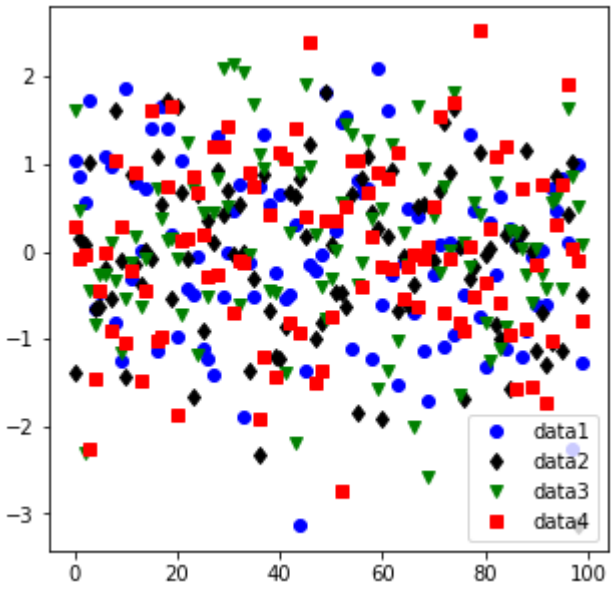


除了set_linestyle之外，还有一些参数可以注意一下，linewidth 表示线的宽度；color 表示线的颜色。

第二，点的风格。我们通过下面这个例子来介绍。

```
fig, ax = plt.subplots(figsize=(5, 5))
ax.plot(data1, 'bo', label='data1')
ax.plot(data2, 'kd', label='data2')
ax.plot(data3, 'gv', label='data3')
ax.plot(data4, 'rs', label='data4')
ax.legend()
```

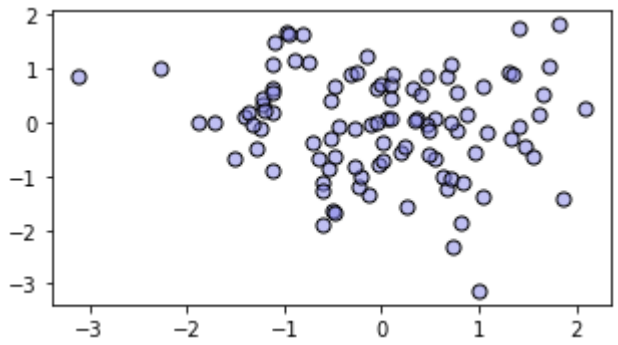
```
<matplotlib.legend.Legend at 0x7fee83ac6250>
```



第三，颜色的表示。在绘制图象时，颜色的选取是很关键的一个问题。我们先来看下面这个例子。

```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.scatter(data1, data2, s=50, facecolor=(0.5, 0.5, 0.9, 0.5), edgecolor='k')
```

```
<matplotlib.collections.PathCollection at 0x7fee84506520>
```

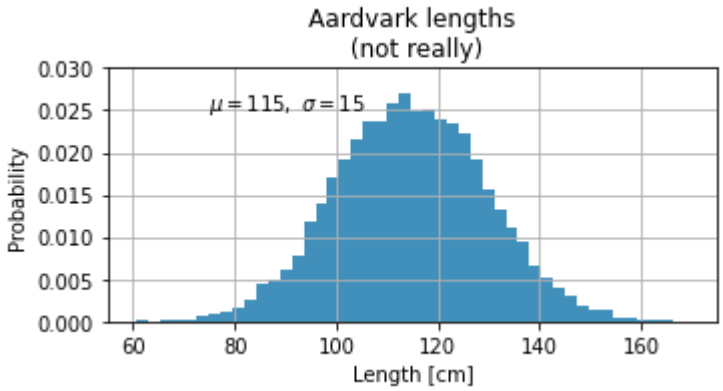


其中，facecolor表示点的内部颜色，edgecolor表示点的边缘颜色。这里颜色表示用的是RGBA(red, green, blue, alpha) 的形式，取值均在[0,1]之间。

第四，图象上必要的文字。通常来说，我们在x轴、y轴、标题上可以添加文字，也可以在图片内添加必要的说明性问题，如下例所示。

```
mu, sigma = 115, 15
x = mu + sigma * np.random.randn(10000)
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
# the histogram of the data
n, bins, patches = ax.hist(x, 50, density=True, facecolor='C0', alpha=0.75)

ax.set_xlabel('Length [cm]')
ax.set_ylabel('Probability')
ax.set_title('Aardvark lengths\n (not really)')
ax.text(75, .025, r'$\mu=115,\ \sigma=15$')
ax.axis([55, 175, 0, 0.03])
ax.grid(True)
```

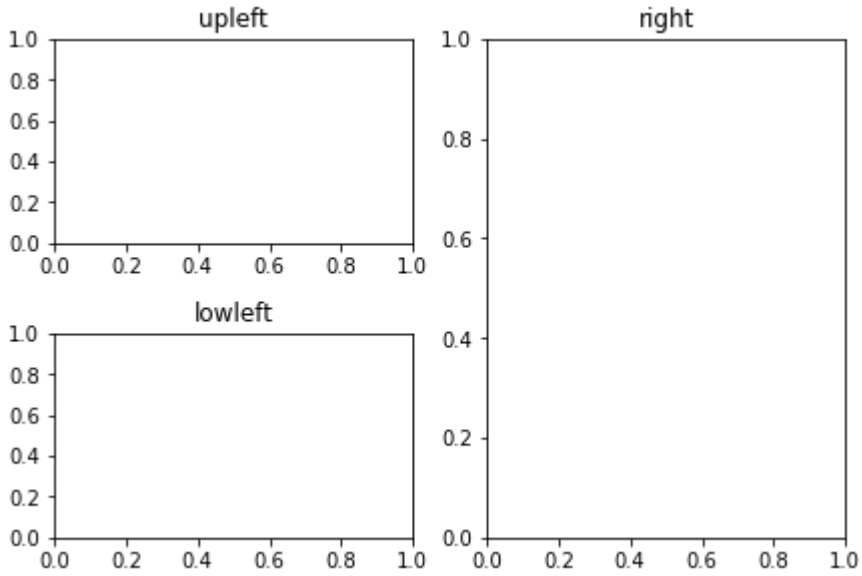


第五，图片的分割。在实际绘图过程中，我们需要把图象去进行合理规划才能绘制合适的图象，以下代码可以将一个画布分为3个区域。

```
fig, axd = plt.subplot_mosaic([['upleft', 'right'],
                               ['lowleft', 'right']], layout='constrained')
```

```
axd['upleft'].set_title('upleft')
axd['lowleft'].set_title('lowleft')
axd['right'].set_title('right')
```

```
Text(0.5, 1.0, 'right')
```

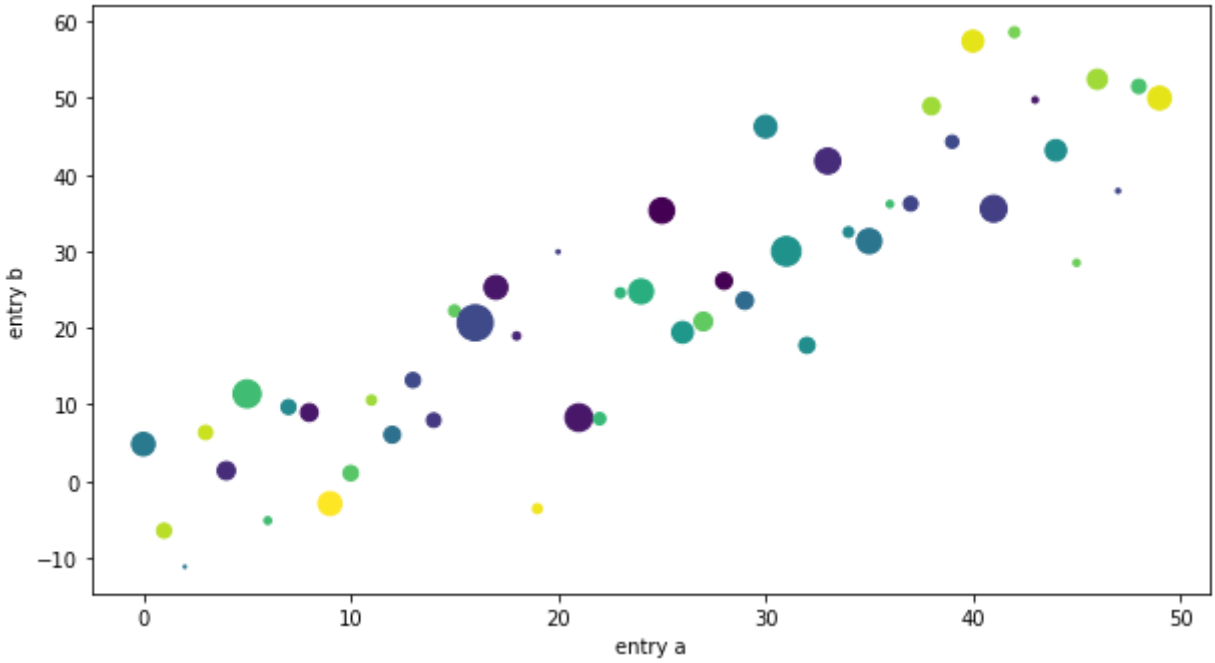


Task4：如何保存图象？

绘制图象后，我们可能需要在其他场合中使用该图象，如何进行保存是一个关键问题。这里我们介绍一种图象格式的保存方式。

```
np.random.seed(19680801)  # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100


fig, ax = plt.subplots(figsize=(10, 5.4))
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
fig.savefig('./demo.pdf')
```



除了课程中介绍的一些绘图基本技巧，感兴趣的同学可以参考以下资料进一步学习与了解。

[1] https://matplotlib.org/stable/index.html