数据分析 7 绘图和可视化

信息可视化（也叫绘图）是数据分析中最重要的工作之一。它可能是探索过程的一部分，例如，帮助我们找出异常值、必要的数据转换、得出有关模型的idea等。另外，做一个可交互的数据可视化也许是工作的最终目标。

matplotlib是一个用于创建出版质量图表的桌面绘图包（主要是2D方面）。

matplotlib支持各种操作系统上许多不同的GUI后端，而且还能将图片导出为各种常见的矢量（vector）和光栅（raster）图：PDF、SVG、JPG、PNG、BMP、GIF等。

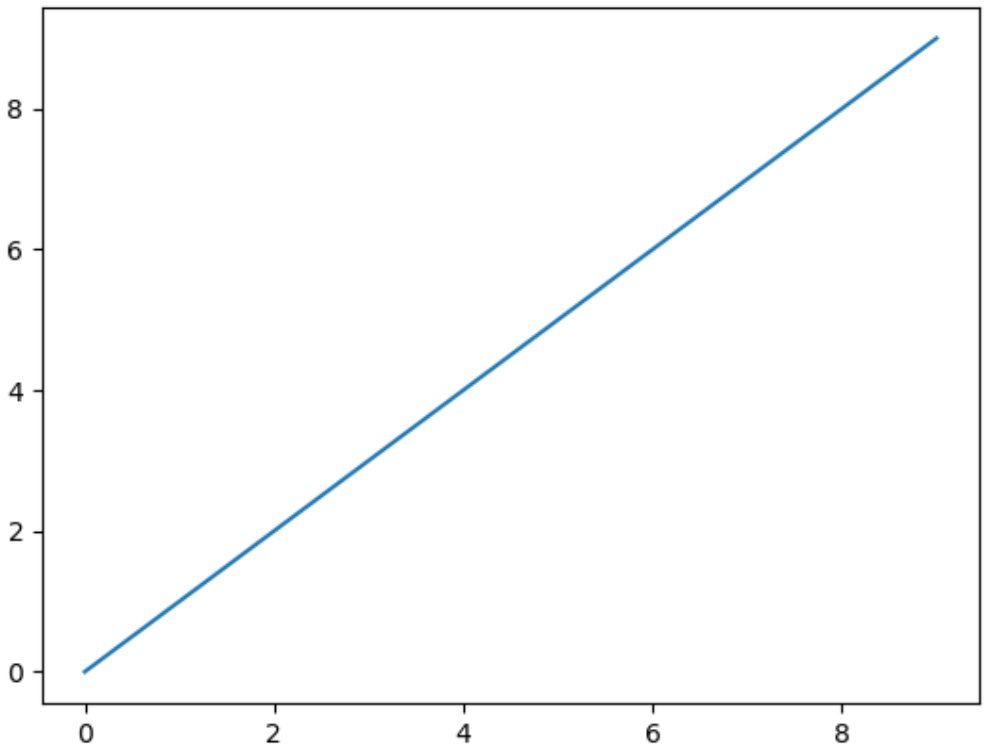
matplotlib API入门

matplotlib的通常引入约定是

```
In [11]: import matplotlib.pyplot as plt
```

在Jupyter中运行%matplotlib notebook 或在IPython中运行%matplotlib

```
In [12]: import numpy as np

In [13]: data = np.arange(10)

In [14]: data
Out[14]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [15]: plt.plot(data)
```



Figure和Subplot

matplotlib的图像都位于Figure对象中。你可以用plt.figure创建一个新的Figure

```
In [16]: fig = plt.figure()
```

不能通过空Figure绘图。必须用add_subplot创建一个或多个subplot才行

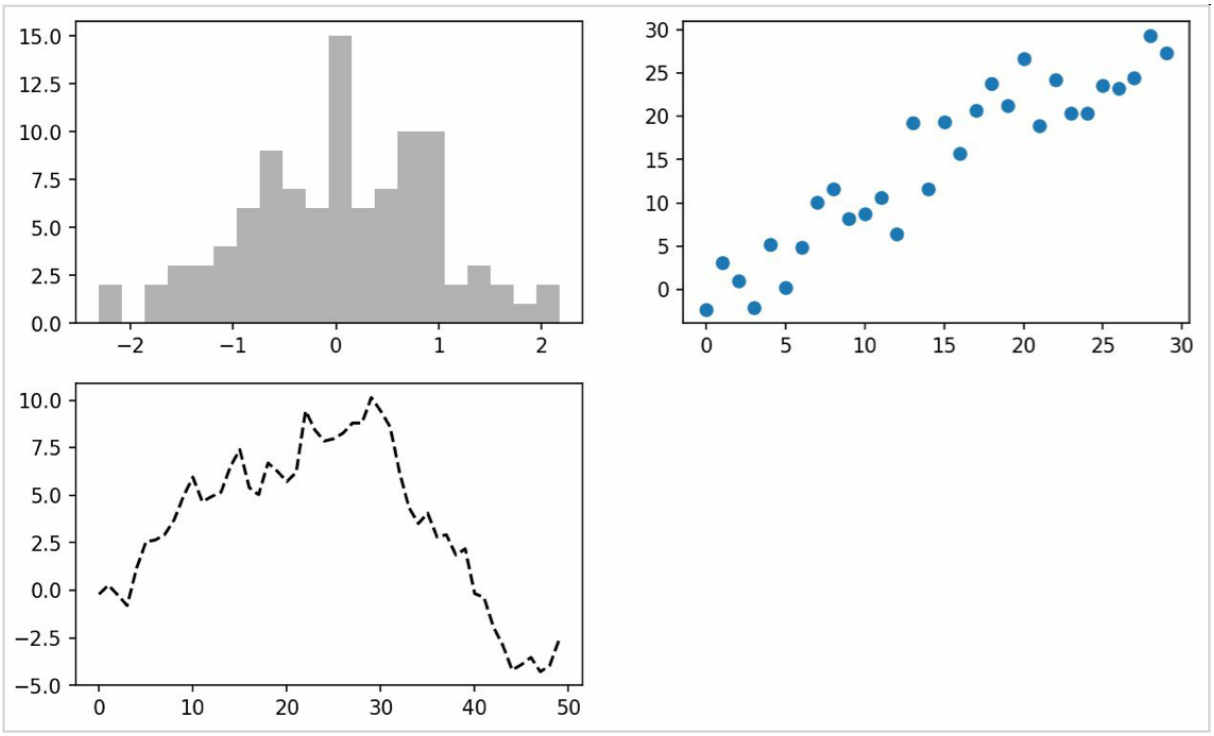
```
# 图像应该是2×2的（即最多4张图），且当前选中的是4个subplot中的第一个（编号从1开始）
In [17]: ax1 = fig.add_subplot(2, 2, 1)
```

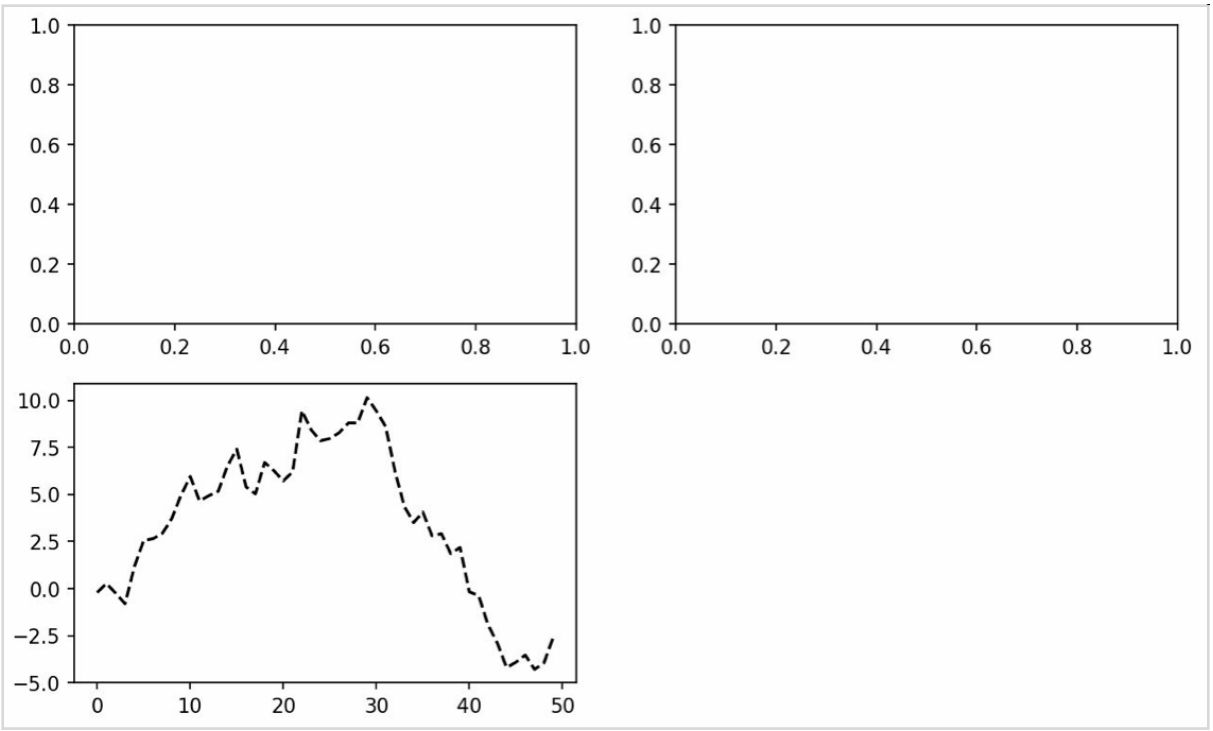
matplotlib会在最后一个用过的subplot（如果没有则创建一个）上进行绘制

```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
# 直方图
plt.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
```

```
ax2 = fig.add_subplot(2, 2, 2)
plt.scatter(np.arange(30), np.arange(30) + 3 *
np.random.randn(30))

ax3 = fig.add_subplot(2, 2, 3)
plt.plot([1.5, 3.5, -2, 1.6])
```

```
In [20]: plt.plot(np.random.randn(50).cumsum(), 'k--')
```





“k–”是一个线型选项，用于告诉matplotlib绘制黑色虚线图

plt.subplots，它可以创建一个新的Figure，并返回一个含有已创建的subplot对象的NumPy数组

```
In [24]: fig, axes = plt.subplots(2, 3)

In [25]: axes
Out[25]:
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fb626374048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb62625db00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb6262f6c88>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb6261a36a0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb626181860>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fb6260fd4e0>]],
dtype
=object)
```

调整subplot周围的间距

默认情况下，matplotlib会在subplot外围留下一定的边距，并在subplot之间留下一定的间距。间距跟图像的高度和宽度有关，因此，如果你调整了图像大小（不管是编程还是手工），间距也会自动调整。利用Figure的subplots_adjust方法可以轻而易举地修改间距

参数	说明
nrows	subplot的行数
ncols	subplot的列数
sharex	所有subplot应该使用相同的X轴刻度（调节xlim将会影响所有subplot）
sharey	所有subplot应该使用相同的Y轴刻度（调节ylim将会影响所有subplot）
subplot_kw	用于创建各subplot的关键字字典
**fig_kw	创建figure时的其他关键字，如plt.subplots(2,2,figsize=(8,6))

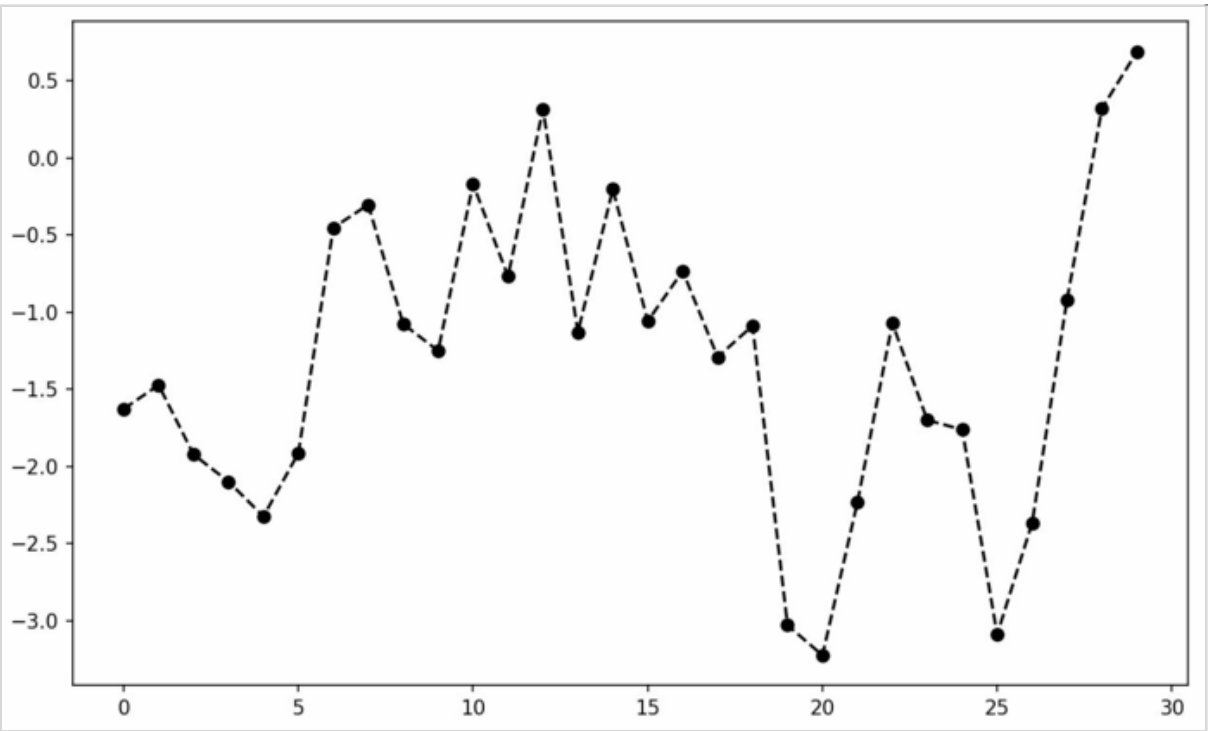
wspace和hspace用于控制宽度和高度的百分比，可以用作subplot之间的间距。下面是一个简单的例子，将间距收缩到了0

```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(4):
          plt.subplot(2, 2, i + 1)
          plt.hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
```

颜色、标记和线型

matplotlib的plot函数接受一组X和Y坐标，还可以接受一个表示颜色和线型的字符串缩写。

```
In [30]: from numpy.random import randn

In [31]: plt.plot(randn(30).cumsum(), ‘ko—‘)
```



简单方法

```
plt.plot(np.random.randn(30).cumsum(), color='k', linestyle='dashed',
marker='i')
```

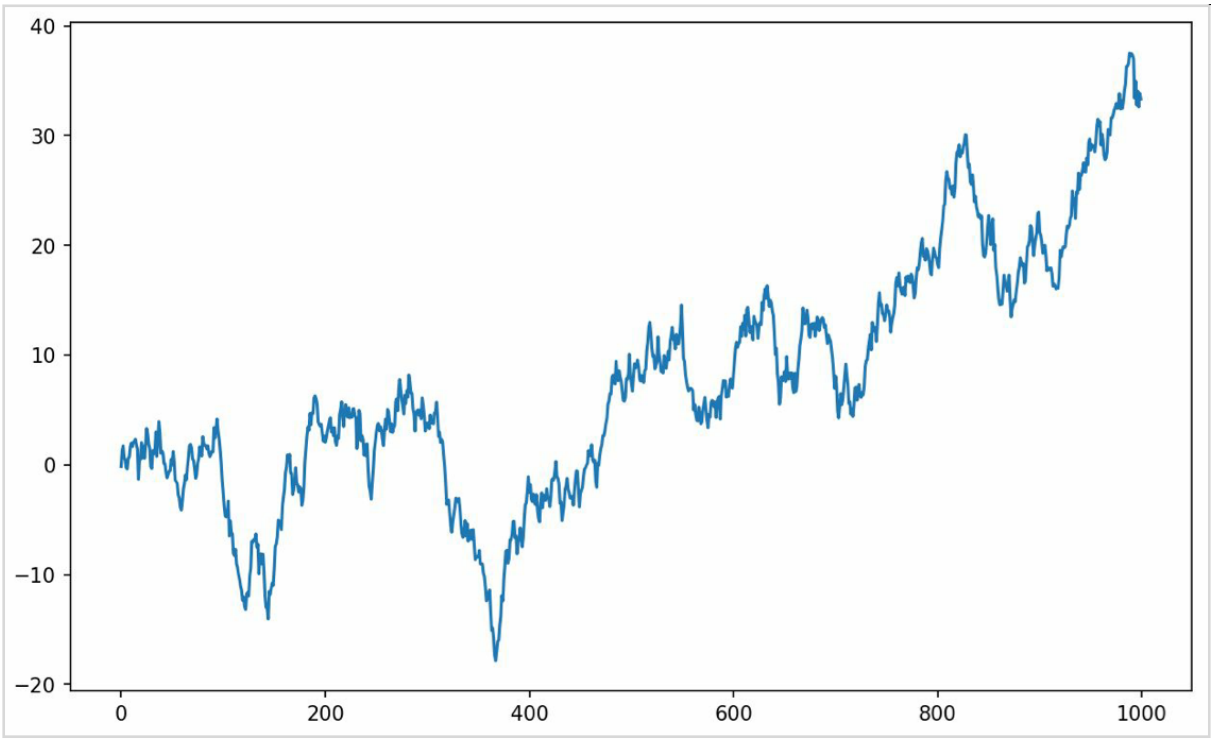
非实际数据点默认是按线性方式插值的。可以通过drawstyle选项修改
在线型图中，非实际数据点默认是按线性方式插值的。可以通过drawstyle选项修改

```
In [33]: data = np.random.randn(30).cumsum()

In [34]: plt.plot(data, 'k--', label='Default')
Out[34]: [<matplotlib.lines.Line2D at 0x7fb624d86160>]

In [35]: plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')
Out[35]: [<matplotlib.lines.Line2D at 0x7fb624d869e8>]
```

设置标题、轴标签、刻度以及刻度标签

```
In [37]: fig = plt.figure()
```

```
In [38]: fig.add_subplot(1, 1, 1)

In [39]: plt.plot(np.random.randn(1000).cumsum())
```



要改变x轴刻度，最简单的办法是使用set_xticks和set_xticklabels。前者告诉matplotlib要将刻度放在数据范围中的哪些位置，默认情况下，这些位置也就是刻度标签。但我们可以通过set_xticklabels将任何其他的值用作标签

```
In [40]: plt.xticks([0, 250, 500, 750, 1000], ['one', 'two', 'three', 'four',
'five'], rotation=30, fontsize='small')

In [42]: plt.title('My first matplotlib plot')
Out[42]: <matplotlib.text.Text at 0x7fb624d055f8>

In [43]: plt.xlabel('Stages')
```

添加图例

```
In [44]: from numpy.random import randn

In [45]: fig = plt.figure(); fig.add_subplot(1, 1, 1)

In [46]: plt.plot(randn(1000).cumsum(), 'k', label='one')
Out[46]: [<matplotlib.lines.Line2D at 0x7fb624bdf860>]

In [47]: plt.plot(randn(1000).cumsum(), 'k--', label='two')
Out[47]: [<matplotlib.lines.Line2D at 0x7fb624be90f0>]

In [48]: plt.plot(randn(1000).cumsum(), 'k.', label='three')
Out[48]: [<matplotlib.lines.Line2D at 0x7fb624be9160>]

# plt.legend()来自动创建图例
plt.legend(loc='best')
```

读取文件并显示图表

```
from datetime import datetime

fig = plt.figure()
fig.add_subplot(1, 1, 1)

data = pd.read_csv('examples/spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']

plt.plot(spx, 'k-')
```

图表保存到文件

```
plt.savefig('figpath.png', dpi=400, bbox_inches='tight')
```

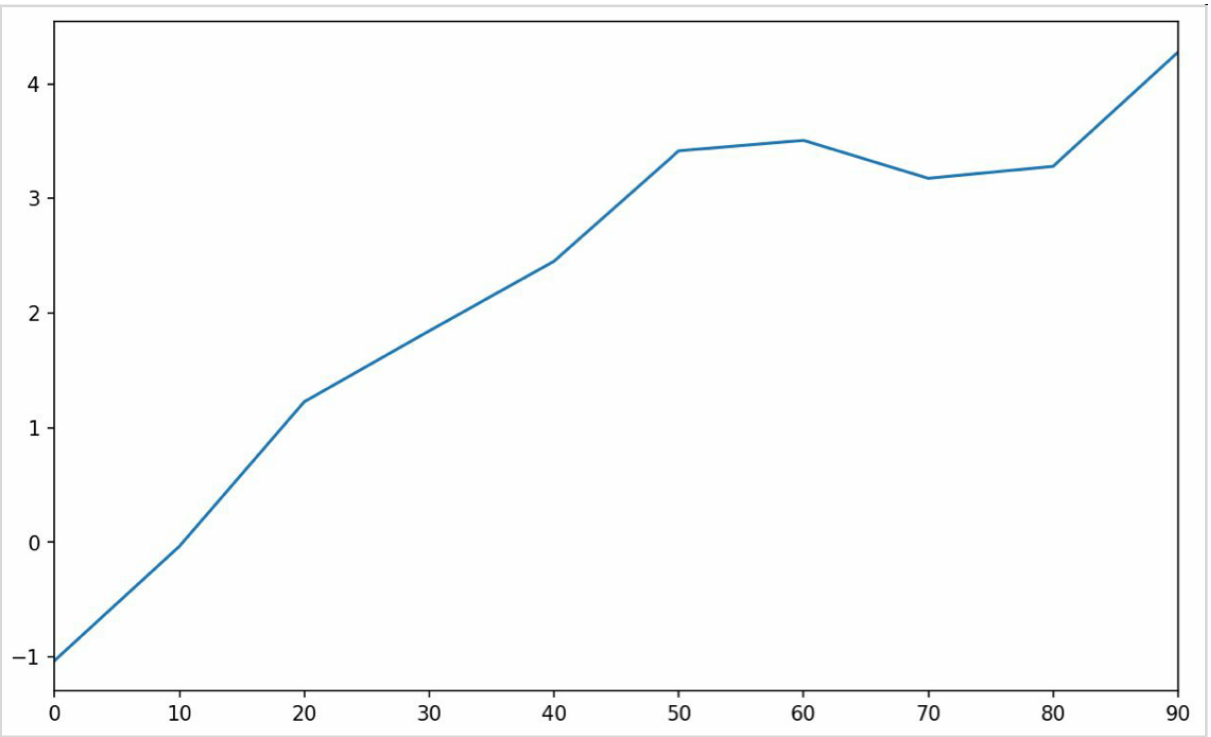
使用pandas和seaborn绘图

Seaborn简化了许多常见可视类型的创建

线型图

Series和DataFrame都有一个用于生成各类图表的plot方法。默认情况下，它们所生成的是线型图

```
In [60]: s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100,
10))

In [61]: s.plot()
```

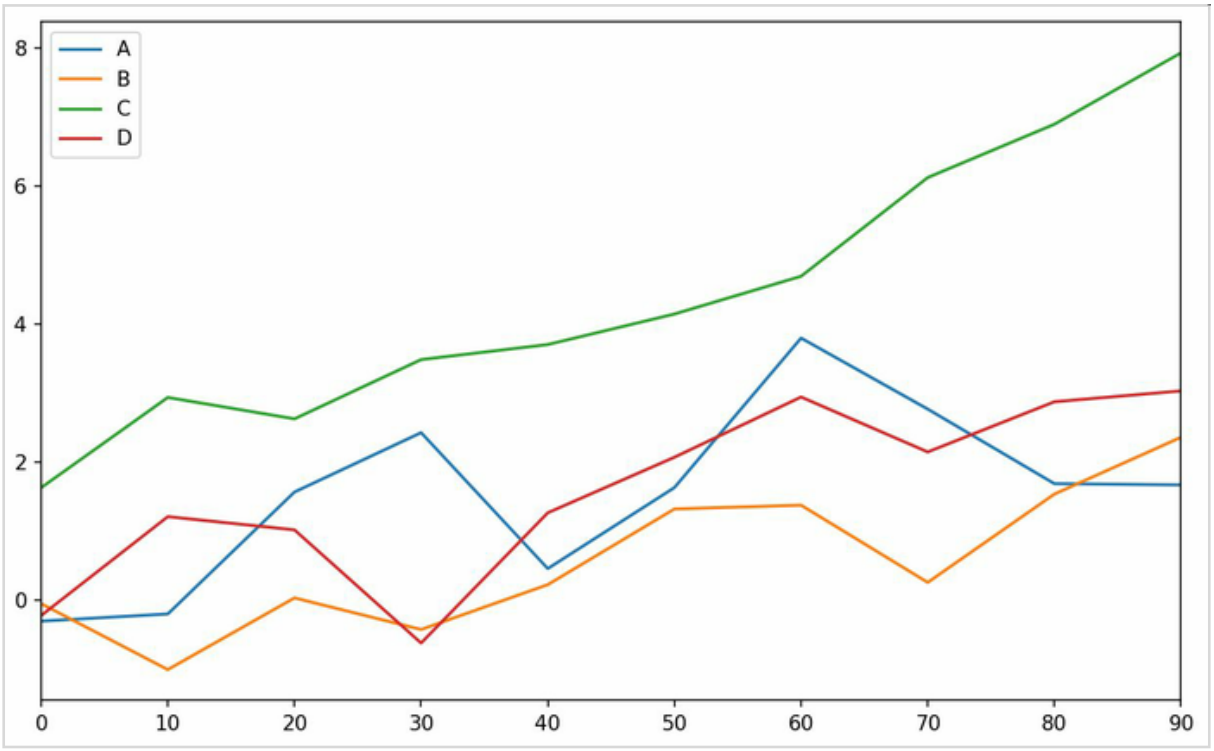


该Series对象的索引会被传给matplotlib，并用以绘制X轴。可以通过use_index=False禁用该功能。X轴的刻度和界限可以通过xticks和xlim选项进行调节，Y轴就用yticks和ylim

DataFrame的plot方法会在一个subplot中为各列绘制一条线，并自动创建图例

```
In [62]: df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
   ....:                   columns=['A', 'B', 'C', 'D'],
   ....:                   index=np.arange(0, 100, 10))
```

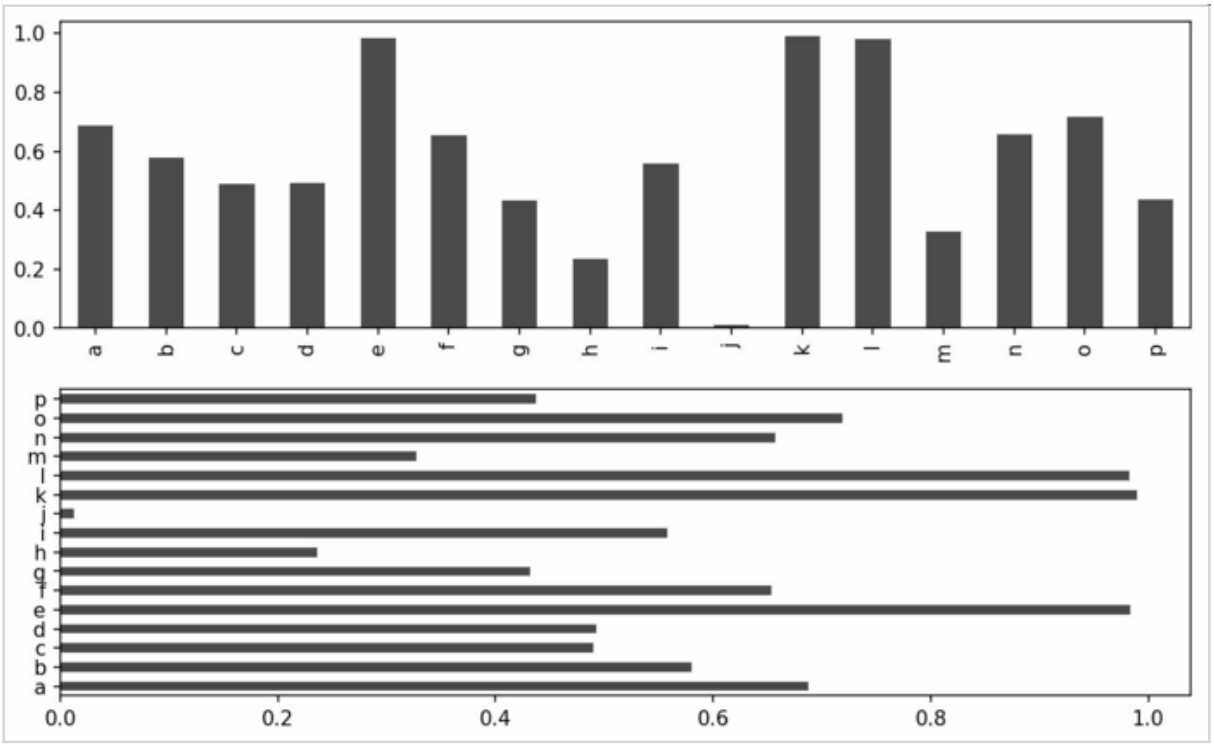
```
In [63]: df.plot()
```



柱状图

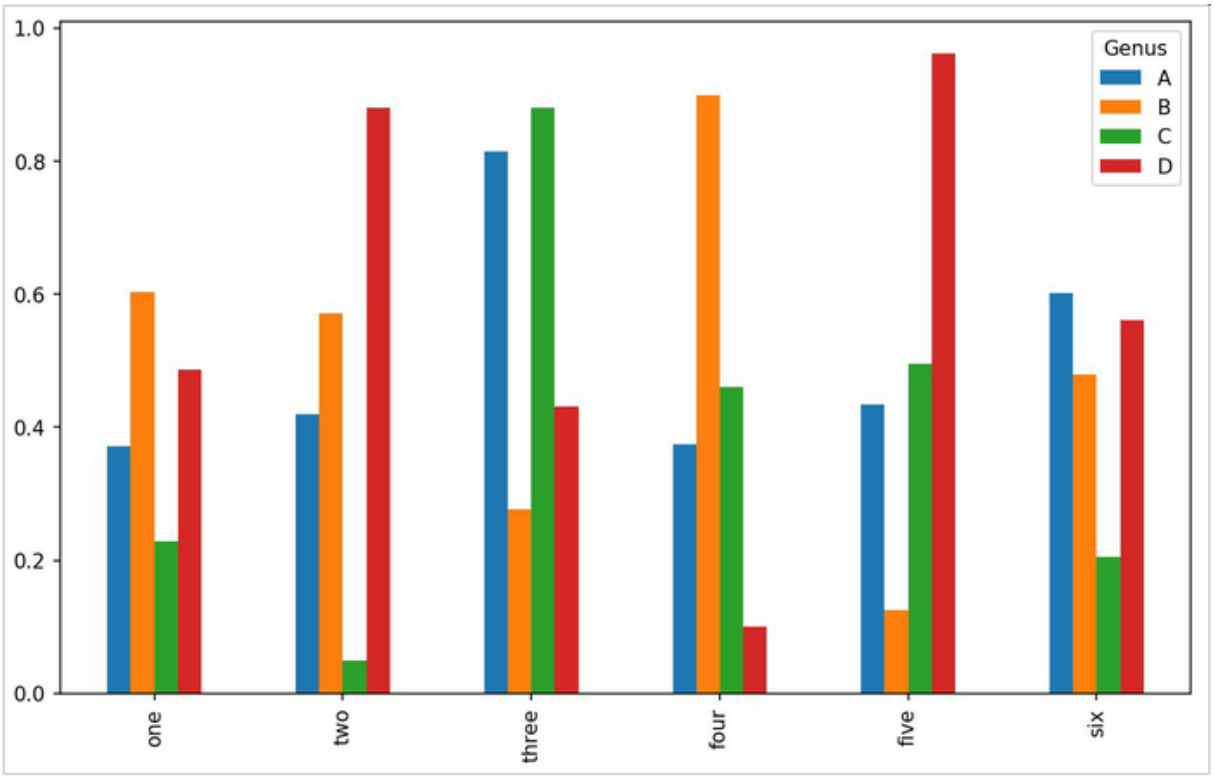
plot.bar()和plot.barh()分别绘制水平和垂直的柱状图。这时，Series和DataFrame的索引将会被用作X（bar）或Y（barh）刻度

```
In [64]: fig, axes = plt.subplots(2, 1)

In [65]: data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))

In [66]: data.plot.bar(ax=axes[0], color='k', alpha=0.7)
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb62493d470>

In [67]: data.plot.barh(ax=axes[1], color='k', alpha=0.7)
```



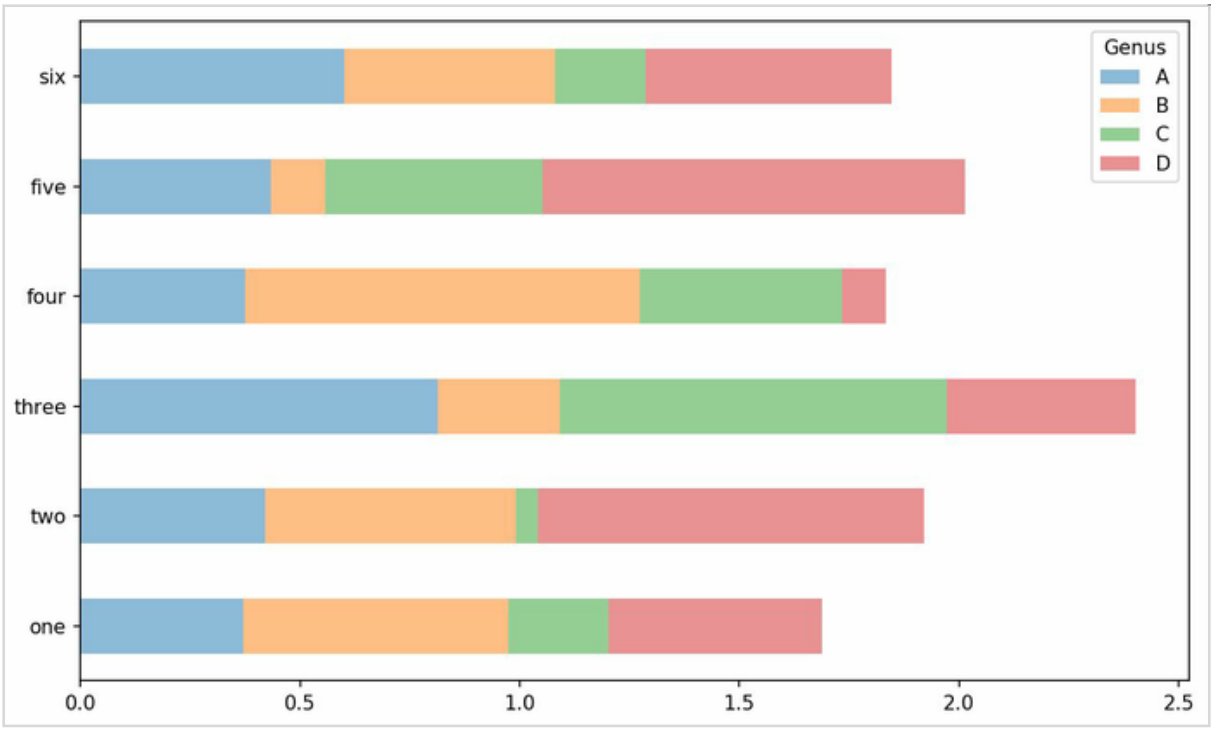
对于DataFrame，柱状图会将每一行的值分为一组，并排显示

```
In [69]: df = pd.DataFrame(np.random.rand(6, 4),
   ....:                   index=['one', 'two', 'three', 'four', 'five',
'six'],
   ....:                   columns=pd.Index(['A', 'B', 'C', 'D'],
name='Genus'))

In [70]: df
Out[70]:
Genus         A         B         C         D
one    0.370670  0.602792  0.229159  0.486744
two    0.420082  0.571653  0.049024  0.880592
three  0.814568  0.277160  0.880316  0.431326
four   0.374020  0.899420  0.460304  0.100843
five   0.433270  0.125107  0.494675  0.961825
six    0.601648  0.478576  0.205690  0.560547

In [71]: df.plot.bar()
```



设置stacked=True即可为DataFrame生成堆积柱状图，这样每行的值就会被堆积在一起

```
In [73]: df.plot.barh(stacked=True, alpha=0.5)
```



柱状图有一个非常不错的用法：利用value_counts图形化显示Series中各值的出现频率，比如

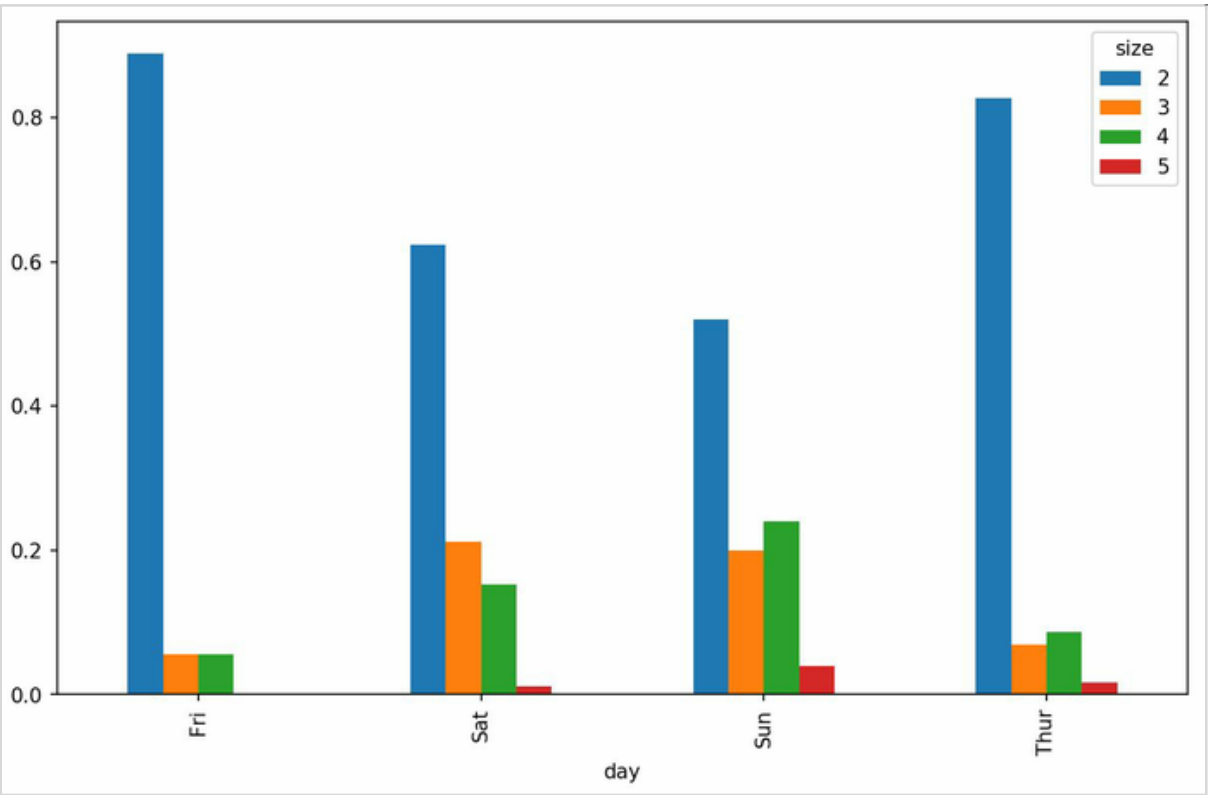
s.value_counts().plot.bar()

以有关小费的数据集为例, 假设我们想要做一张堆积柱状图以展示每天各种聚会规模的数据点的百分比。用read_csv将数据加载进来，然后根据日期和聚会规模创建一张交叉表

```
In [75]: tips = pd.read_csv('examples/tips.csv')

In [76]: party_counts = pd.crosstab(tips['day'], tips['size'])

In [77]: party_counts
Out[77]:
size  1   2   3   4  5  6
day
Fri   1  16   1   1  0  0
Sat   2  53  18  13  1  0
Sun   0  39  15  18  3  1
Thur  1  48   4   5  1  3

# Not many 1- and 6-person parties
In [78]: party_counts = party_counts.loc[:, 2:5]
```

然后进行规格化，使得各行的和为1，并生成图表

```
In [79]: party_pcts = party_counts.div(party_counts.sum(1), axis=0)

In [80]: party_pcts
Out[80]:
size         2         3         4         5
day
Fri   0.888889  0.055556  0.055556  0.000000
Sat   0.623529  0.211765  0.152941  0.011765
Sun   0.520000  0.200000  0.240000  0.040000
Thur  0.827586  0.068966  0.086207  0.017241

In [81]: party_pcts.plot.bar()
```

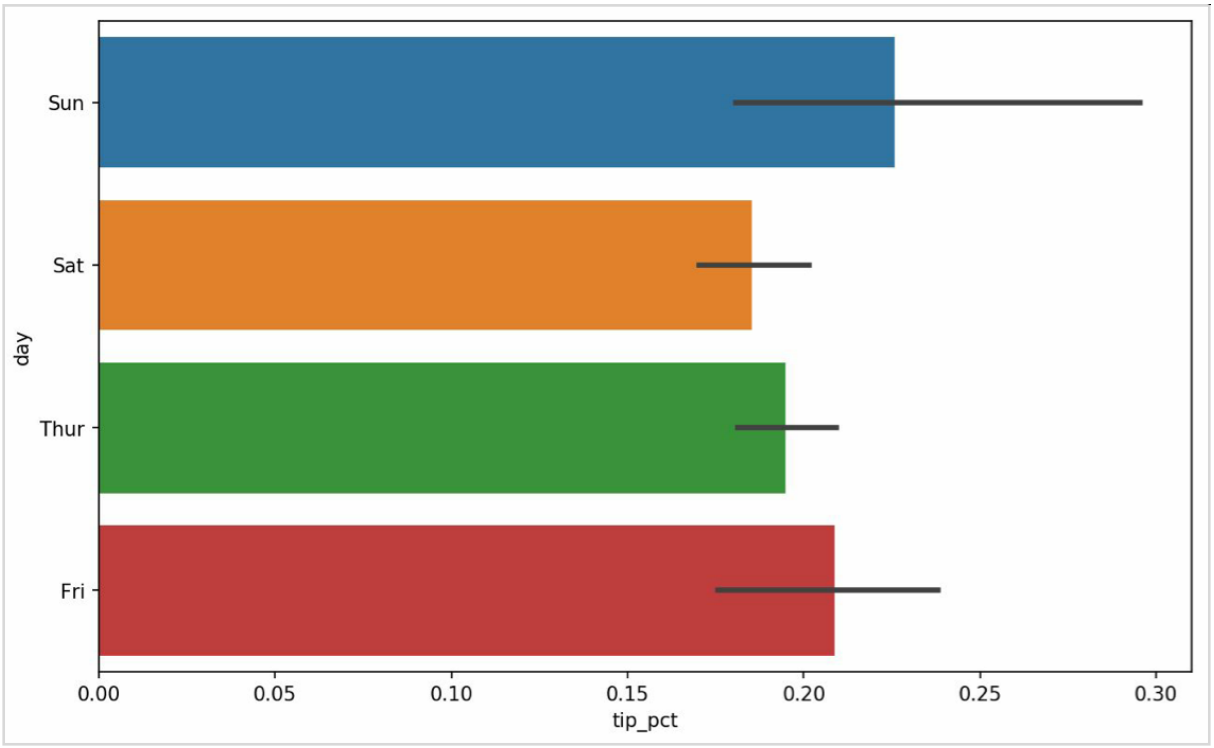


通过该数据集就可以看出，聚会规模在周末会变大

使用seaborn可以减少工作量。用seaborn来看每天的小费比例

```
In [83]: import seaborn as sns

In [84]: tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])

In [85]: tips.head()
Out[85]:
   total_bill   tip smoker  day    time  size   tip_pct
0       16.99  1.01     No  Sun  Dinner     2  0.063204
1       10.34  1.66     No  Sun  Dinner     3  0.191244
2       21.01  3.50     No  Sun  Dinner     3  0.199886
3       23.68  3.31     No  Sun  Dinner     2  0.162494
4       24.59  3.61     No  Sun  Dinner     4  0.172069

In [86]: sns.barplot(x='tip_pct', y='day', data=tips, orient='h')
```

绘制在柱状图上的黑线代表95%置信区间

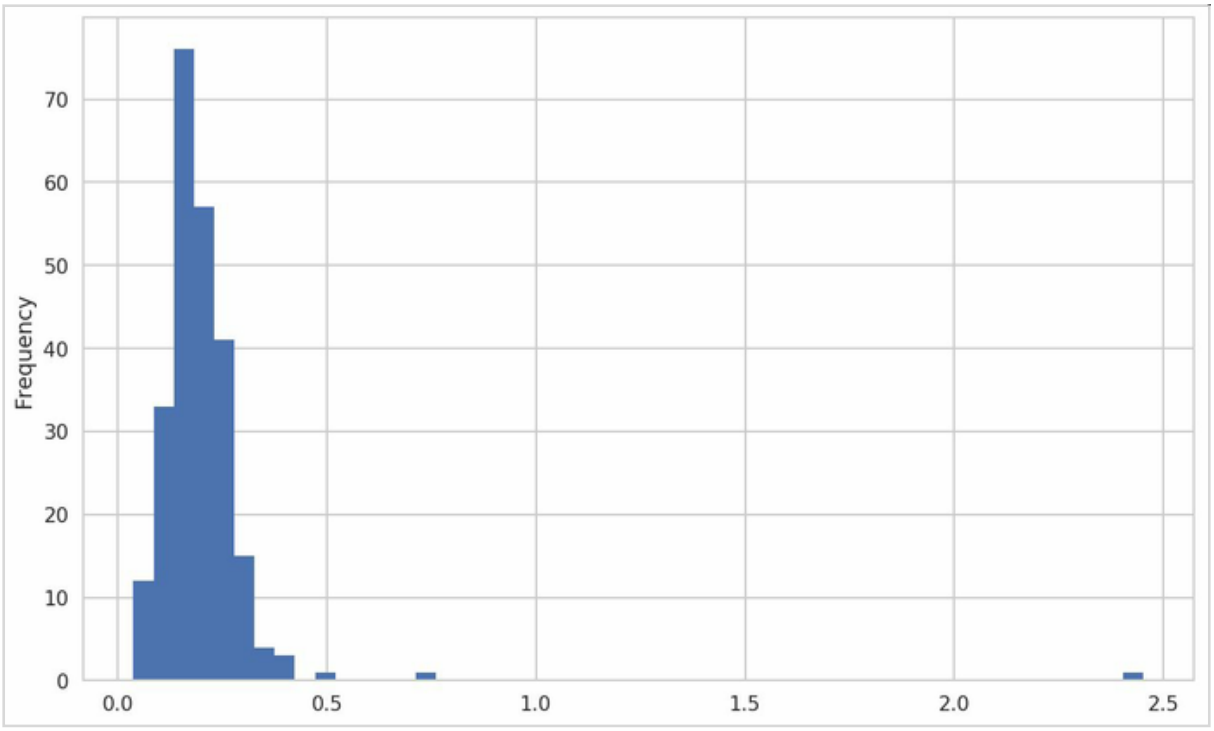



```
In [88]: sns.barplot(x='tip_pct', y='day', hue='time', data=tips, orient='h')
```

直方图和密度图

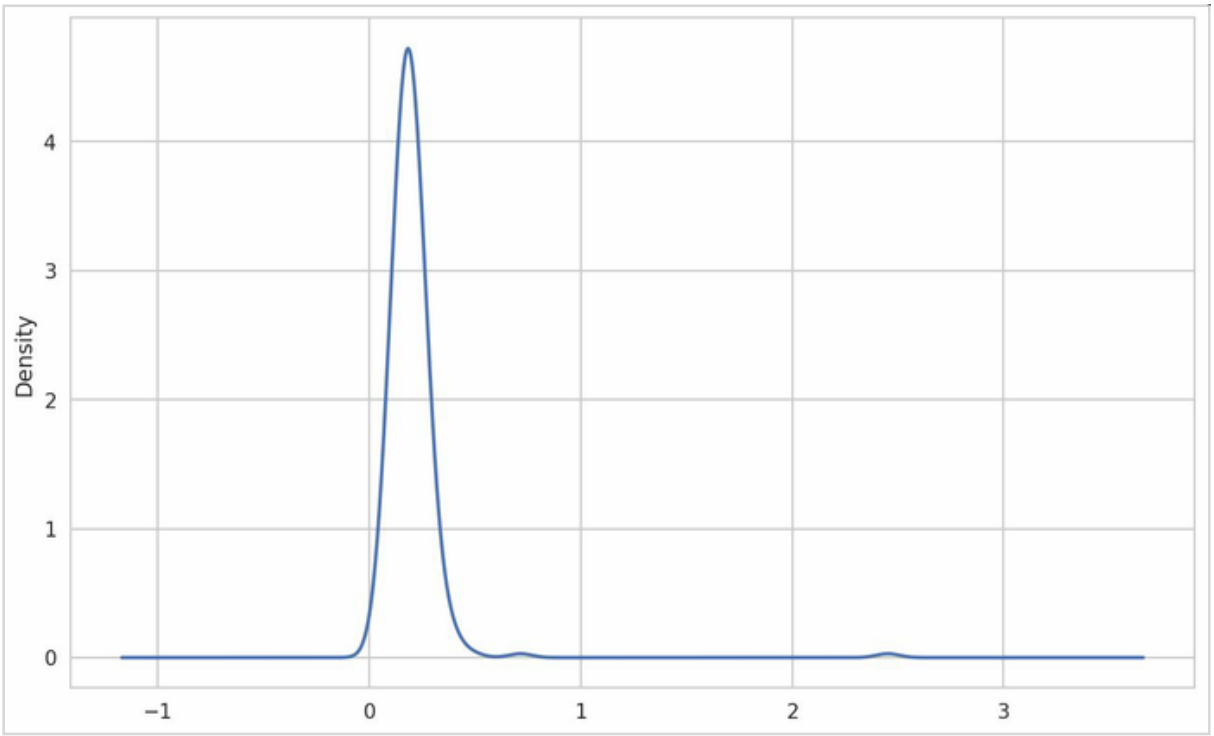
直方图（histogram）是一种可以对值频率进行离散化显示的柱状图。数据点被拆分到离散的、间隔均匀的面元中，绘制的是各面元中数据点的数量。再以前面那个小费数据为例，通过在Series使用plot.hist方法，我们可以生成一张“小费占消费总额百分比”的直方图

```
In [92]: tips[‘tip_pct’].plot.hist(bins=50)
```



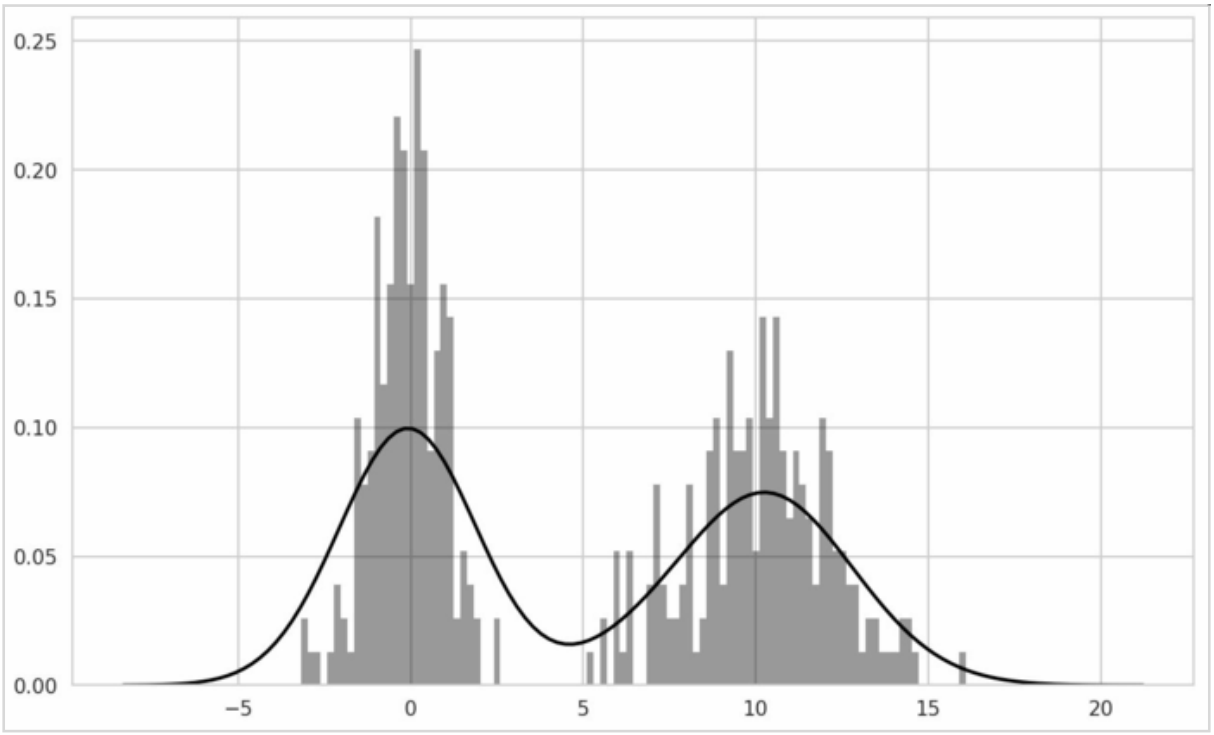
与此相关的一种图表类型是密度图，它是通过计算“可能会产生观测数据的连续概率分布的估计”而产生的。一般的过程是将该分布近似为一组核（即诸如正态分布之类的较为简单的分布）。因此，密度图也被称作KDE（Kernel Density Estimate，核密度估计）图

```
In [94]: tips['tip_pct'].plot.density()
```



用seaborn绘制

```
In [96]: comp1 = np.random.normal(0, 1, size=200)

In [97]: comp2 = np.random.normal(10, 2, size=200)

In [98]: values = pd.Series(np.concatenate([comp1, comp2]))

In [99]: sns.distplot(values, bins=100, color='k')
```



点图或散布图是观察两个一维数据序列之间的关系的有效手段

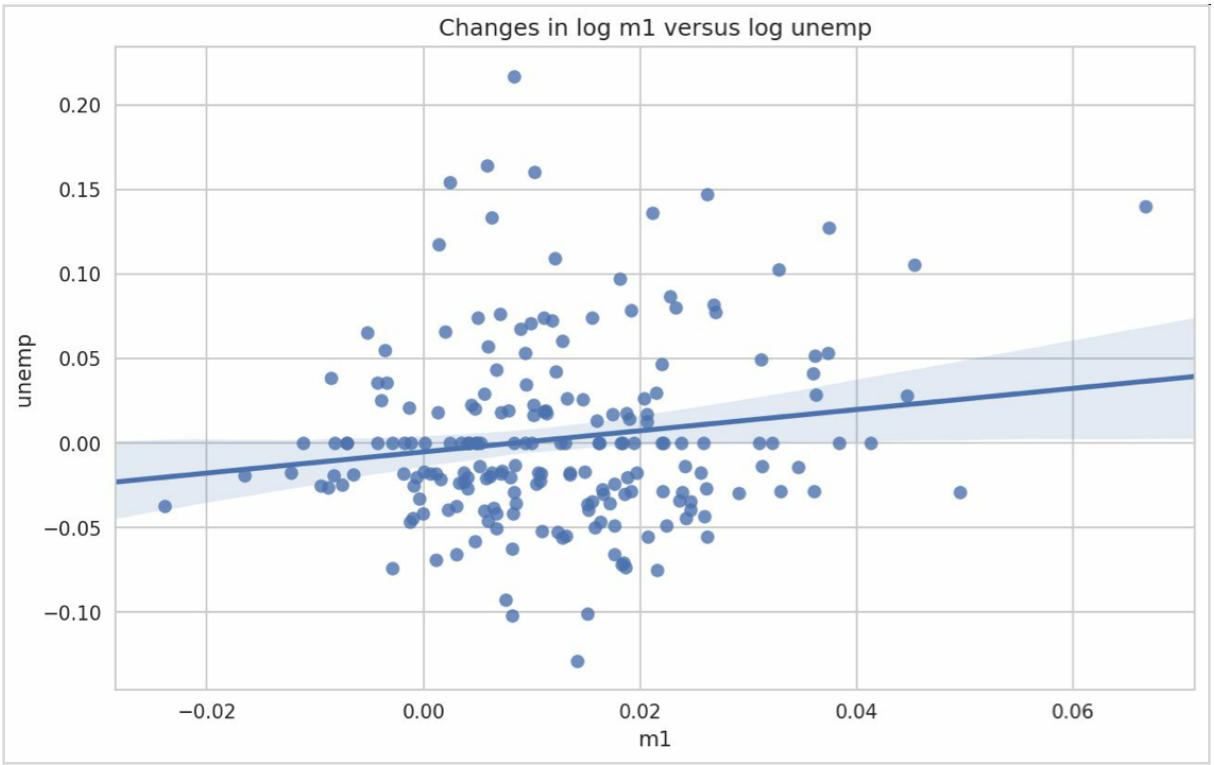
加载了来自statsmodels项目的macrodata数据集，选择了几个变量，然后计算对数差

```
In [100]: macro = pd.read_csv('examples/macrodata.csv')

In [101]: data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]

In [102]: trans_data = np.log(data).diff().dropna()
```

```
In [103]: trans_data[-5:]
Out[103]:
          cpi        m1  tbilrate     unemp
198 -0.007904  0.045361 -0.396881  0.105361
199 -0.021979  0.066753 -2.277267  0.139762
200  0.002340  0.010286  0.606136  0.160343
201  0.008419  0.037461 -0.200671  0.127339
202  0.008894  0.012202 -0.405465  0.042560
```

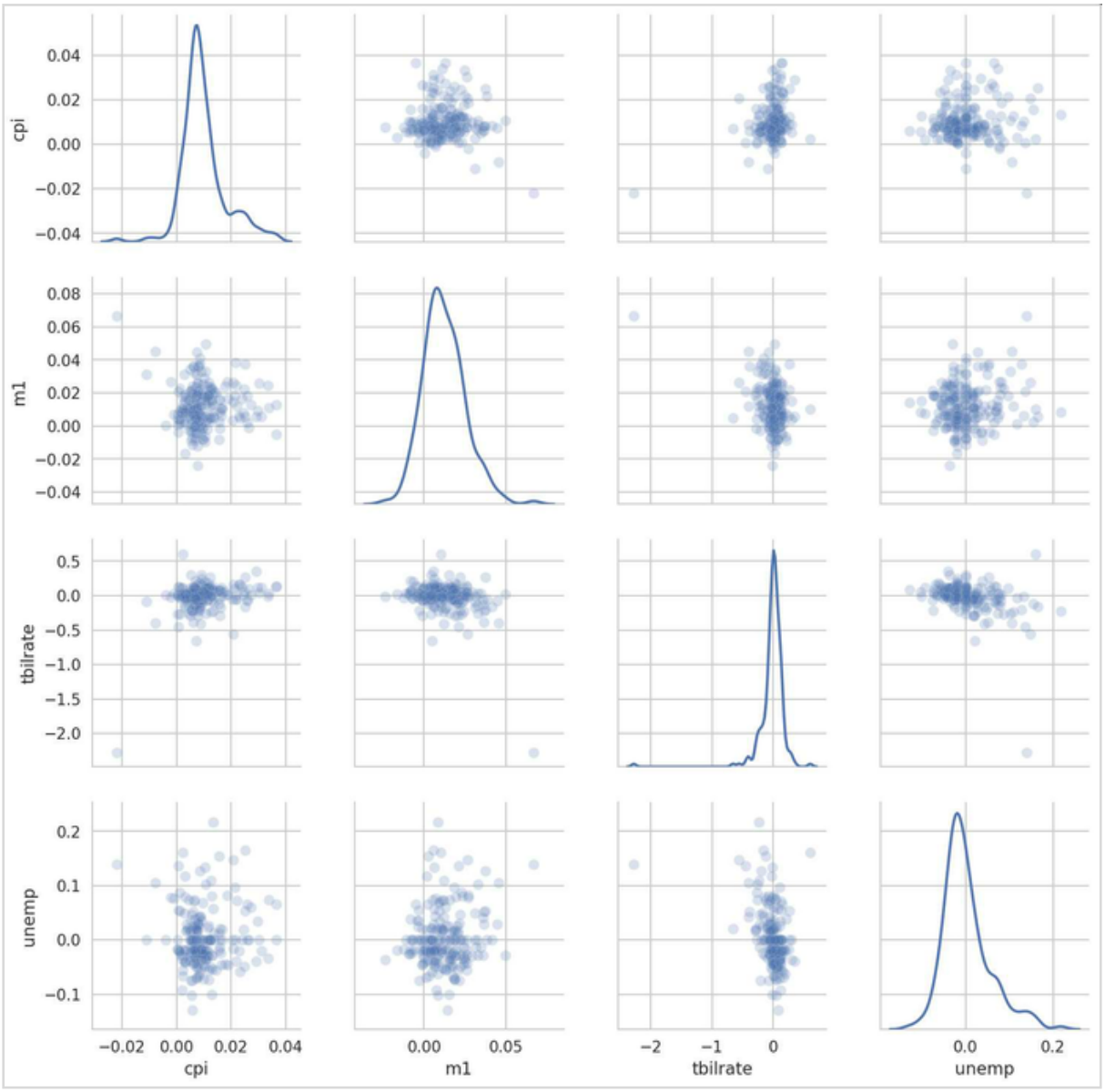
使用seaborn的regplot方法，它可以做一个散布图，并加上一条线性回归的线

```
In [105]: sns.regplot('m1', 'unemp', data=trans_data)
Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb613720be0>

In [106]: plt.title('Changes in log %s versus log %s' % ('m1', 'unemp'))
```



seaborn提供了一个便捷的pairplot函数，它支持在对角线上放置每个变量的直方图或密度估计

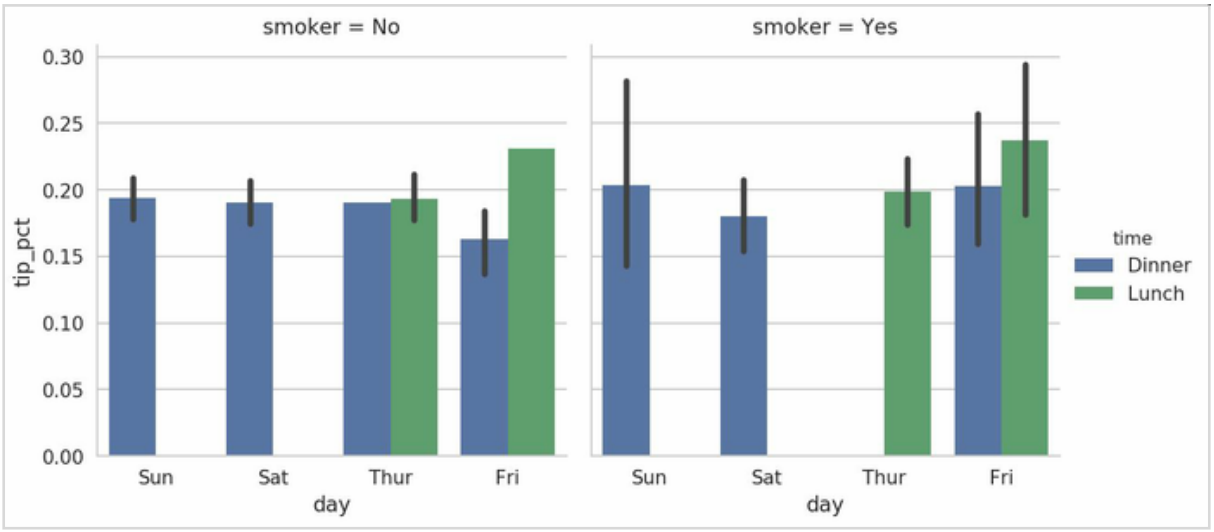
```
In [107]: sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.2})
```



分面网格（facet grid）和类型数据

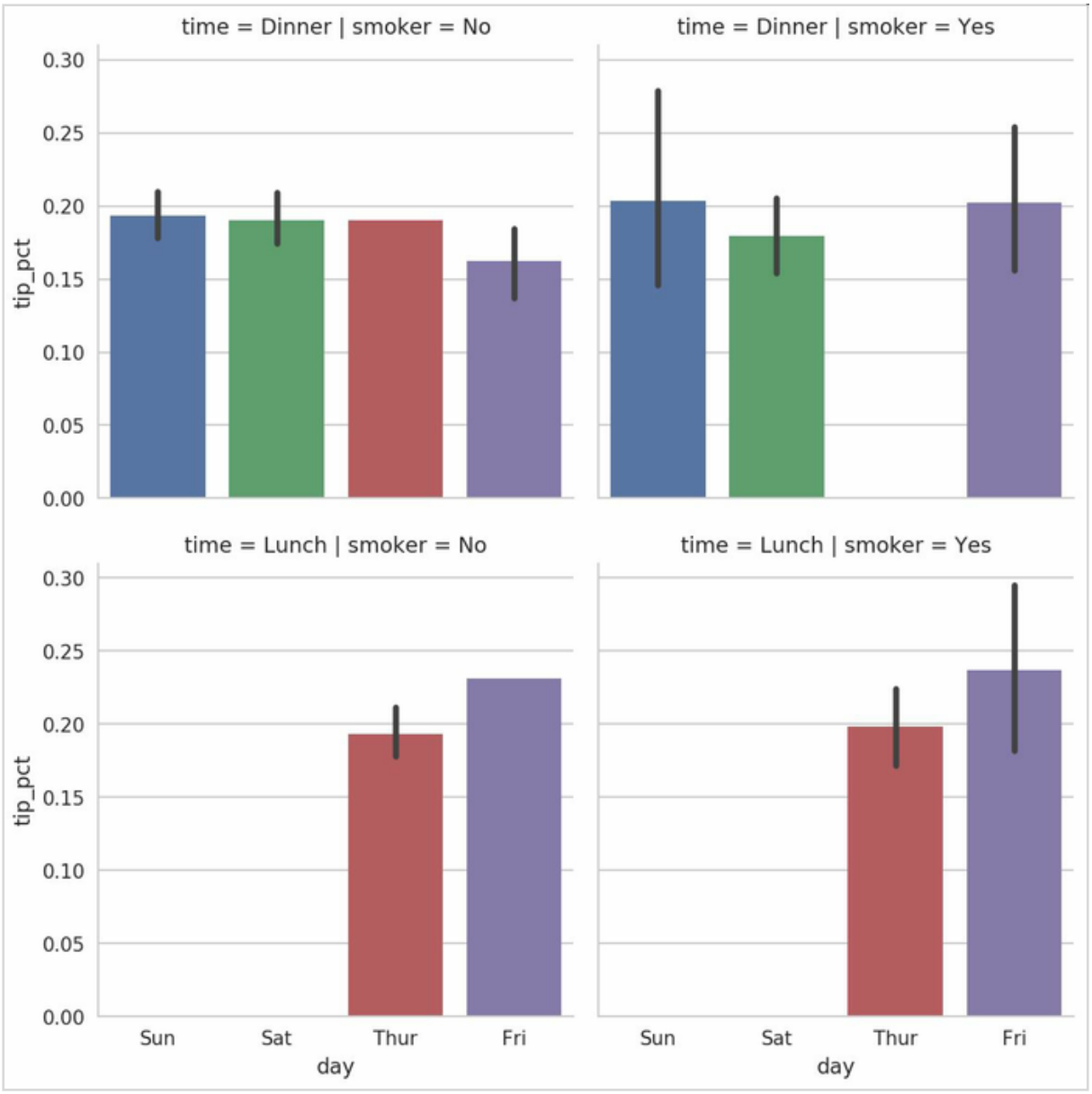
数据集有额外的分组维度, seaborn有一个有用的内置函数factorplot，可以简化制作多种分面图

```
In [108]: sns.factorplot(x='day', y='tip_pct', hue='time', col='smoker',
   .....:                kind='bar', data=tips[tips.tip_pct < 1])
```



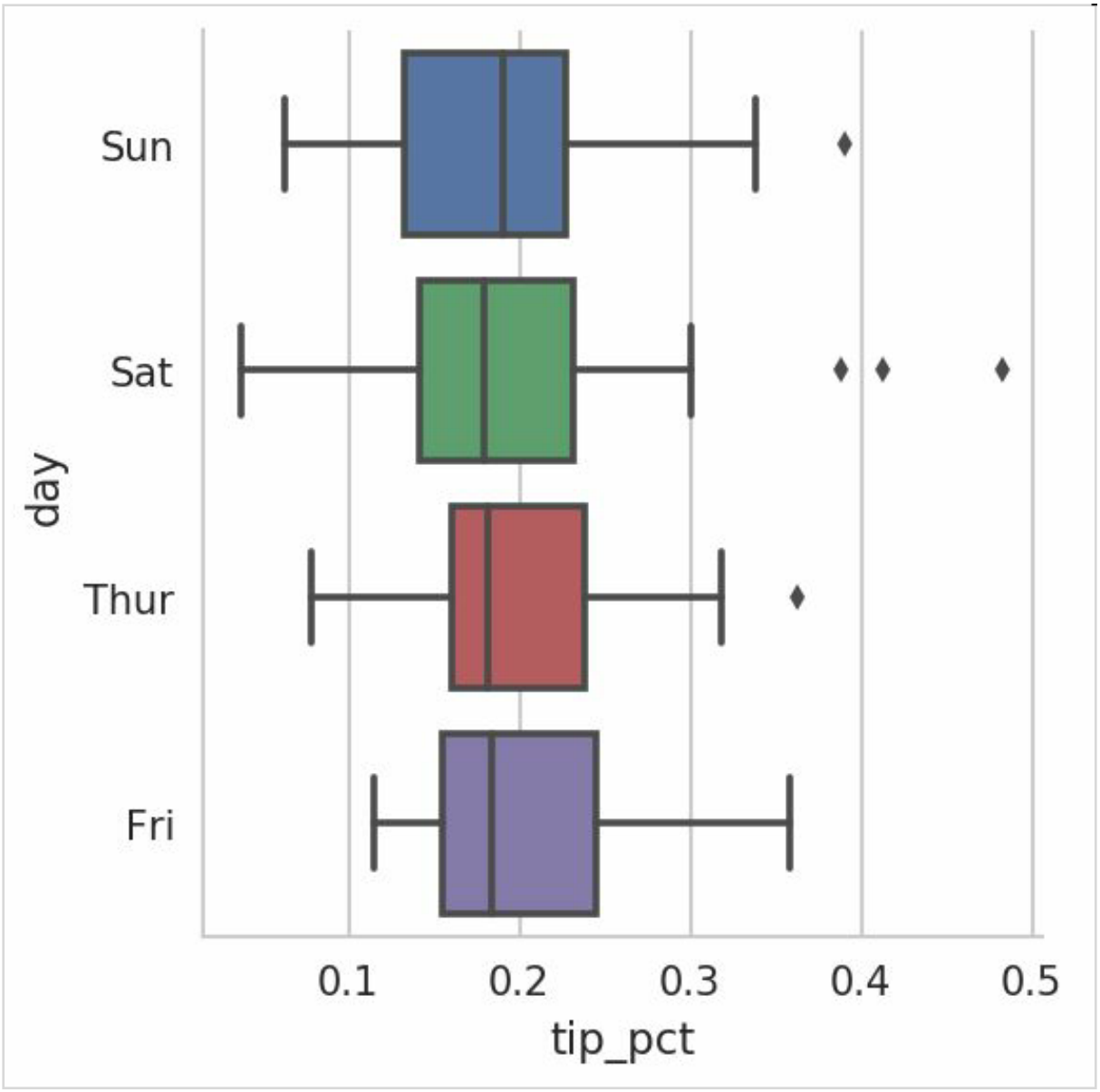
除了在分面中用不同的颜色按时间分组，我们还可以通过给每个时间值添加一行来扩展分面网格

```
In [109]: sns.factorplot(x='day', y='tip_pct', row='time',
   .....:                col='smoker',
   .....:                kind='bar', data=tips[tips.tip_pct < 1])
```



factorplot支持其它的绘图类型，你可能会用到。例如，盒图（它可以显示中位数，四分位数，和异常值）就是一个有用的可视化类型

```
In [110]: sns.factorplot(x='tip_pct', y='day', kind='box',
   .....:                data=tips[tips.tip_pct < 0.5])
```


Sun
Sat
Thur
Fri
day
0.1
0.2
0.3
0.4
0.5
tip_pct