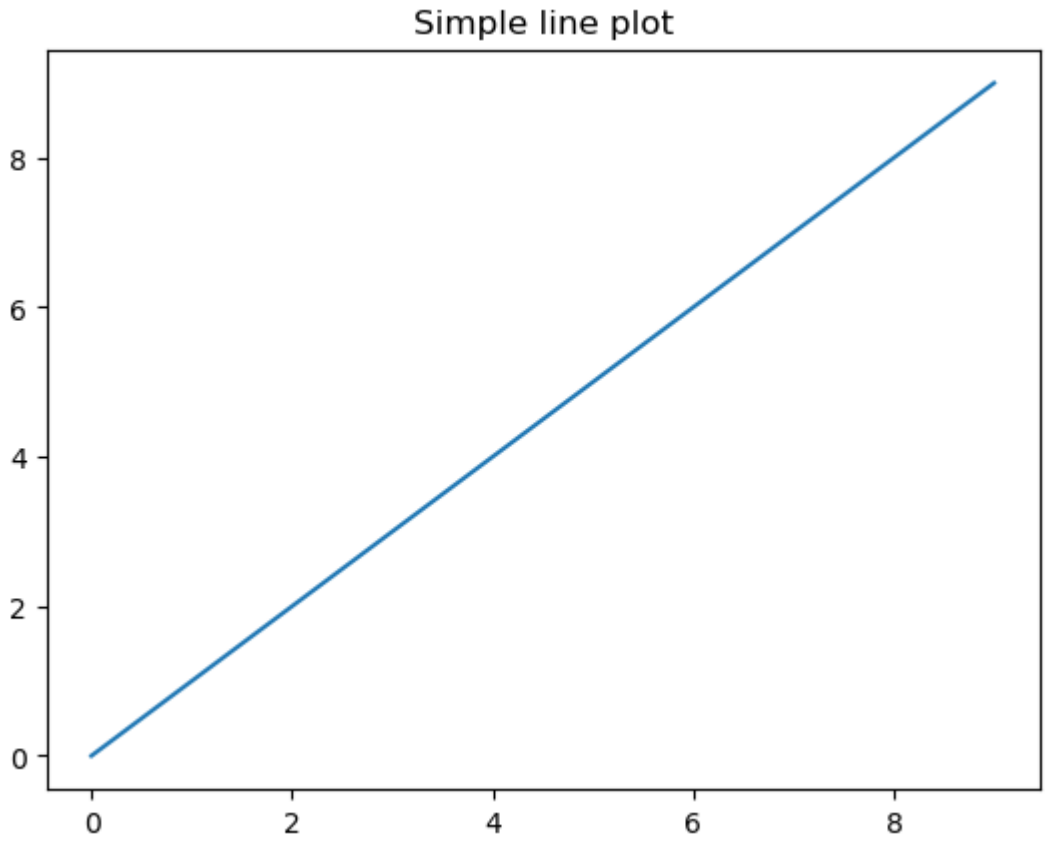
In [1]:
```
#魔法指令用于在jupyter notebook中显示图形
%matplotlib notebook
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [3]:
```
data = np.arange(10)
data
#array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
plt.plot(data)
plt.title("Simple line plot")
```

Out[3]:
```
Text(0.5, 1.0, 'Simple line plot')
```



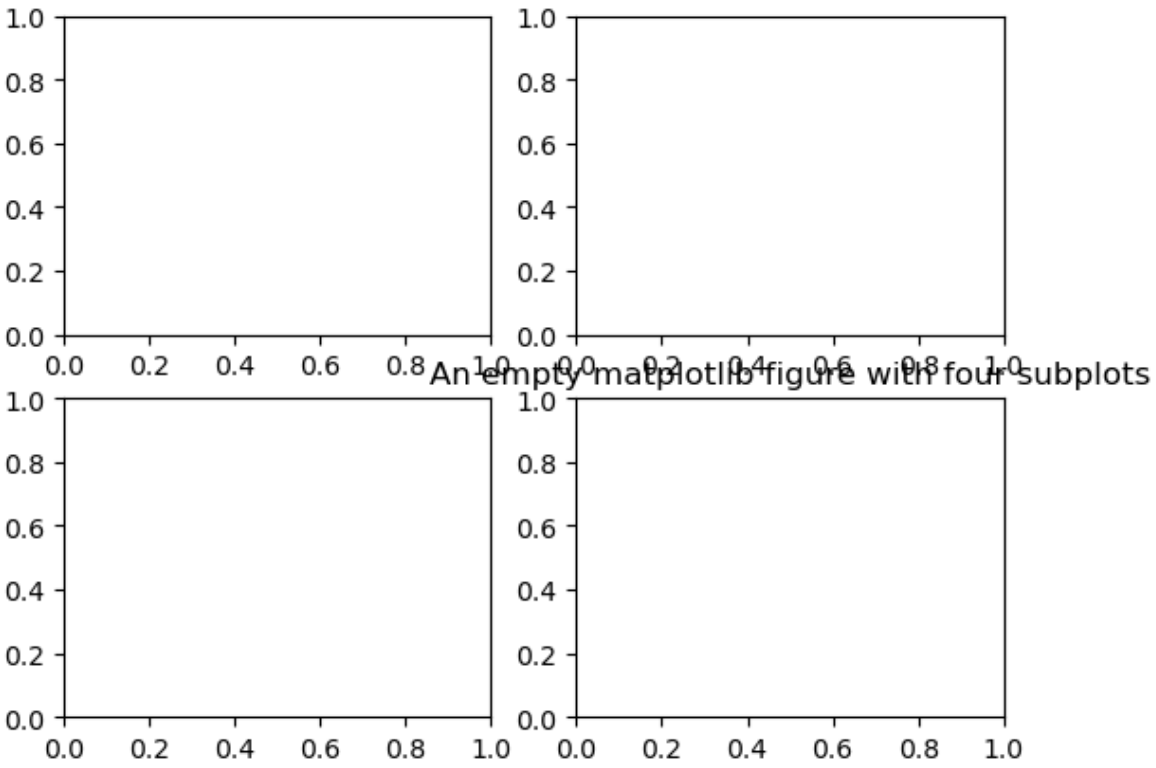
Ch9: Plotting and Visualization 绘图与可视化

9.1 简明matplotlib API入门 A Brief matplotlib API Primer

9.1.1 图片与子图 Figures and Subplots

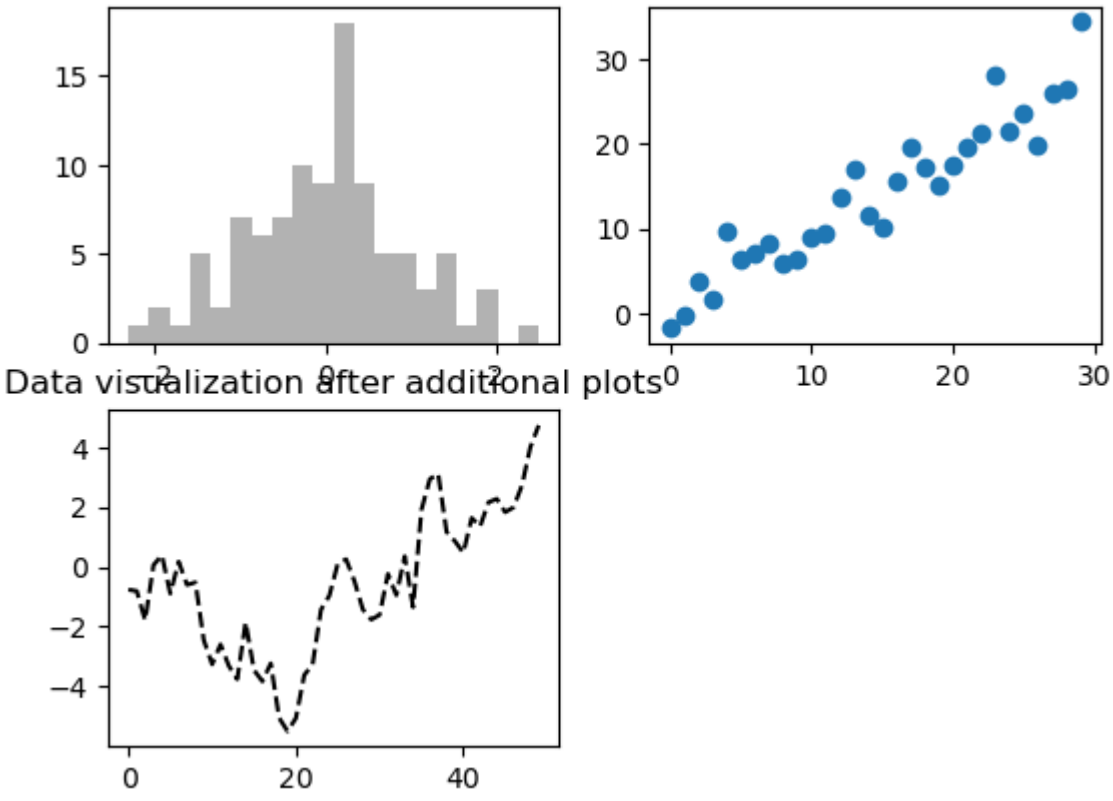
In [4]:
```
#matplotlib 绘制的图位于图片(Figure)中。可以使用plt.figure生成一个新的图片:
#一个带有三个子图的空白matplotlib图片

fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)   #图片应该是2*2（最多4个图形），并且选择了四个图形
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)   #最多放4个图片
plt.title("An empty matplotlib figure with four subplots")
```

Out[4]:
```
Text(0.5, 1.0, 'An empty matplotlib figure with four subplots')
```



In [5]:
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)

#当输入绘图命令plt.plot([1.5, 3.5, -2, 1.6]), matplotlib会在最后一个图片和子图上进
#fig.add_subplot返回的对象是axes subplot对象。

#生成一个随机数序列并进行累计求和，然后使用黑色虚线绘制该累计求和序列的折线图：
plt.plot(np.random.randn(50).cumsum(), 'k--')

#生成一个长度为100的随机数序列，这些随机数是符合标准正态分布（均值为0，方差为1）的。
#指定直方图的箱数为20，即将数据划分成20个区间。
#使用下划线 _ 接收函数返回的对象，表示不关心该对象，仅仅是为了阻止输出，并非必须的操作
#在名为 ax1 的子图上绘制了一个黑色的直方图，展示了随机数序列的分布情况，其中直方图的边
_ = ax1.hist(np.random.randn(100), bins = 20, color = 'k', alpha = 0.3)

#生成一个长度为30的等差数列。
#生成一个长度为30的随机数序列，每个随机数乘以3，然后加上前面生成的等差数列。
#scatter 函数绘制散点图，其中第一个参数是 x 坐标，第二个参数是 y 坐标
#散点图展示了 x 和 y 坐标之间的关系，其中 x 坐标是等差数列，y 坐标是根据随机数生成的
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
plt.title("Data visualization after additional plots")
```

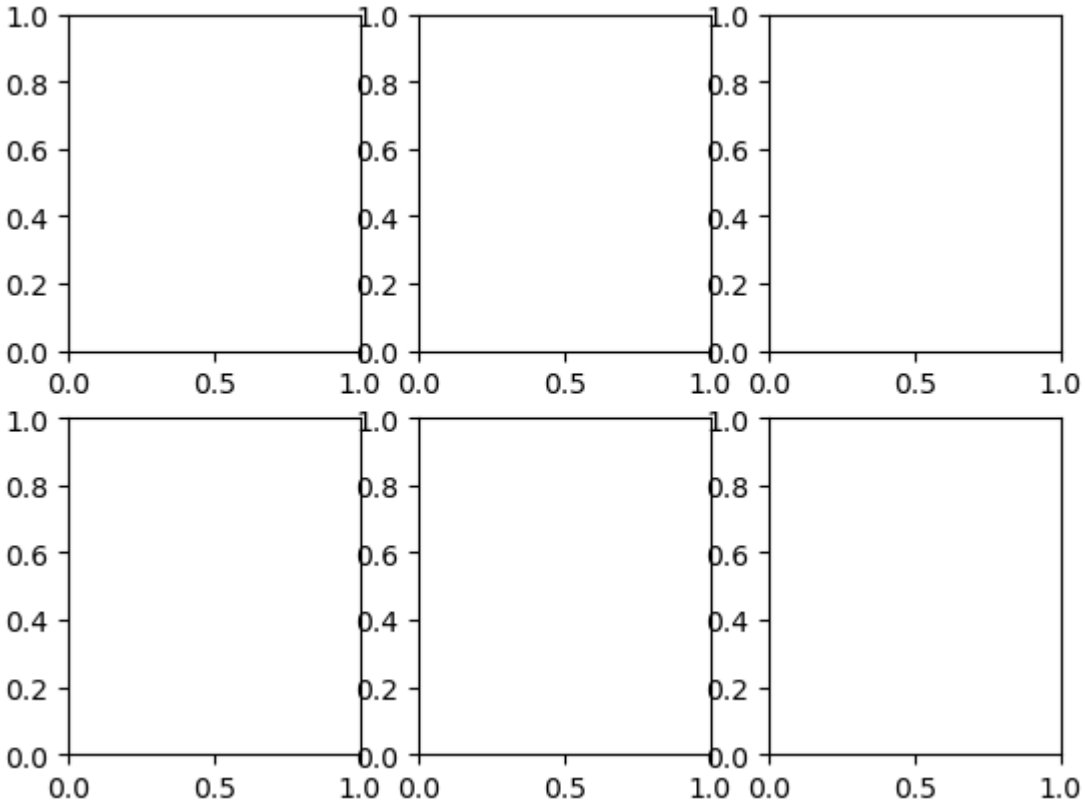
Out[5]:
```
Text(0.5, 1.0, 'Data visualization after additional plots')
```



In [6]:
```
#MATPLOTLIB官方绘图指南：
#https://matplotlib.org/stable/plot_types/index.html
```

In [7]:
```
#展示了如何一次创建多个子图，并将子图存储在数组中，以便后续对每个子图进行个别设置和绘
#使用子网格创建图片，创建一个新的图片，返回包含了已生成子图对象的numpy数组。
#axes 是一个包含所有子图的numpy数组。在这个例子中，axes 是一个2行3列的数组，代表了2
#每个子图都可以通过 axes 数组的索引访问，例如 axes[0, 0] 是第一行第一列的子图，axes
fig, axes = plt.subplots(2,3)
axes

#数组axes可以像二维数组那样方便地进行索引，axes[0, 1]
#使用sharex和sharey表名子图分别拥有相同的x轴或y轴。
#当在相同比例下，sharex和sharey会非常有用。
```

Out[7]:
```
array([[<Axes: >, <Axes: >, <Axes: >],
       [<Axes: >, <Axes: >, <Axes: >]], dtype=object)
```

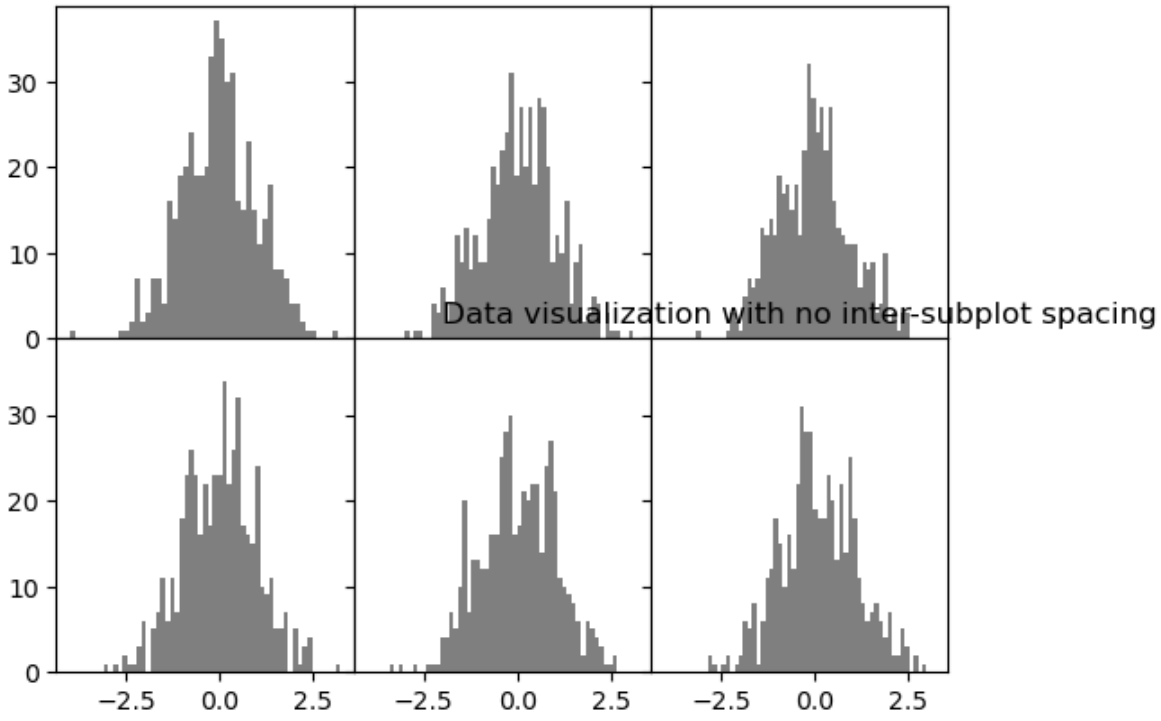


In [8]:
```
#matplotlib可以独立缩放图像界限。
#pyplot.subplots选项
#nrows：子图的行数
#ncols：子图的列数
#sharex：所有子图使用相同的x轴刻度
#sharey：所有子图使用相同的y轴刻度
#subplot_kw：传入add_subplot的关键字参数字典，用于生成子图。
#**fig_kw：在生成图片时，使用额外关键字参数，plt.subplots(2, 2, figsze = (8, 6))
```

9.1.1.1 调整子图周围的间距

In [9]:
```
#把几个子图拼在一起：
#matplotlib会在子图的外部和子图之间留出一定的间距。这个间距是相对于图的高度和宽度来指
#可以使用subplots_adjust方法更改间距，也可以用作顶层函数：
#subplots_adjust(left = None, bottom = None, right = None, top = None, wspace
#wspace和hspace分别控制的是图片的宽度和高度百分比，以用作子图间的间距。可以将间距缩小
#.subplots_adjust(wspace=0, hspace=0)：调整子图之间的间距，将水平和垂直间距都设置为
#在两个嵌套循环通过axes[i, j]访问每个子图，并在每个子图中使用hist函数绘制随机生成的数

fig, axes = plt.subplots(2, 3, sharex = True, sharey = True)
for i in range(2):
    for j in range(3):
        axes[i, j].hist(np.random.randn(500), bins = 50, color = 'k', alpha =
plt.subplots_adjust(wspace = 0, hspace = 0)
plt.title("Data visualization with no inter-subplot spacing")

#轴标签存在重叠的。matplotlib不检查标签是否重叠。因此在类似情况下，需要通过显式刻度位
```

Out[9]: Text(0.5, 1.0, 'Data visualization with no inter-subplot spacing')

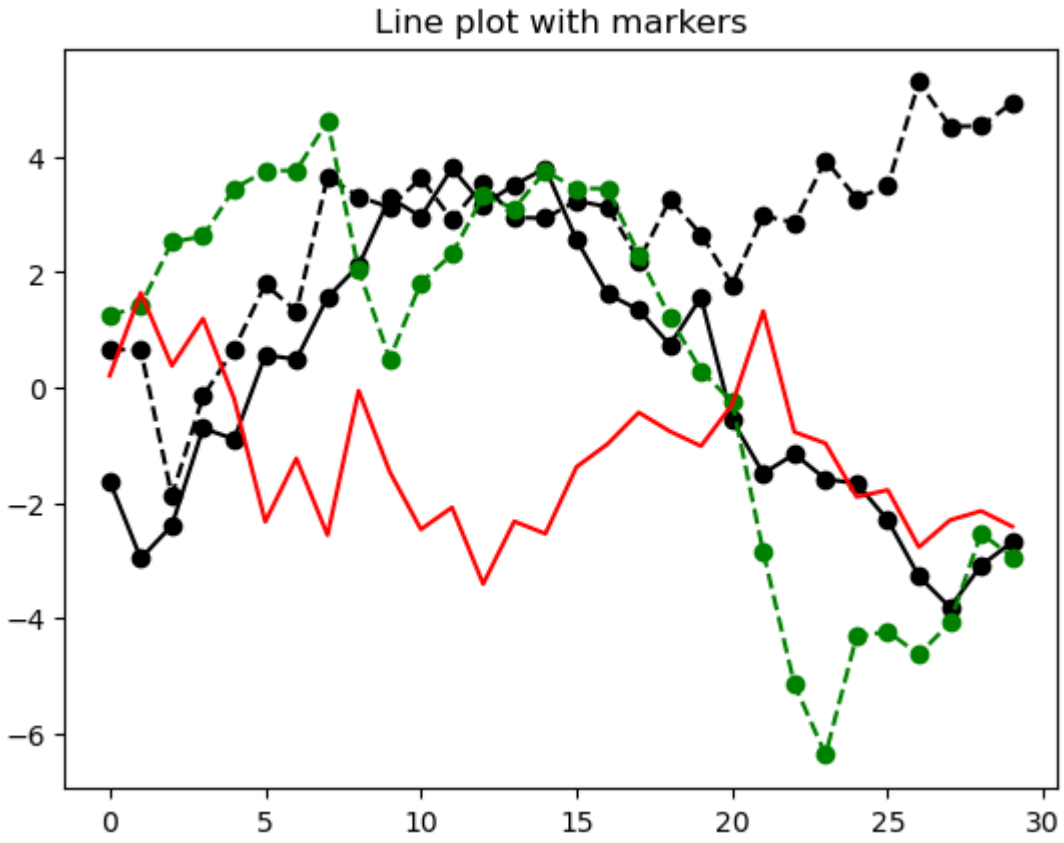


9.1.2 颜色、标记和线类型 Colors, Markers, and Line Styles

In [10]:
```
import matplotlib.pyplot as plt
from numpy.random import randn
# #matplotlib主函数plot接收带有x和y轴数组，以及一些可选的字符串缩写参数来指明颜色和
# #绿色破折号绘制x对y的线
# ax.plot(x, y, 'g--')
# ax.plot(x, y, linestyle = '--', color = 'g')

# #很多颜色缩写被用于常用颜色，但可以指定16进制颜色代码方式来制定颜色（#cecece）。

plt.plot(randn(30).cumsum(), 'ko-')                          #ko-：实线有点
plt.plot(randn(30).cumsum(), 'ko--')                         #ko--：虚线有点
plt.plot(randn(30).cumsum(), 'ko--', color = 'g')            #color = 'g':
plt.plot(randn(30).cumsum(), linestyle = '-', color = 'r')   #color = 'r':
plt.title("Line plot with markers")
```

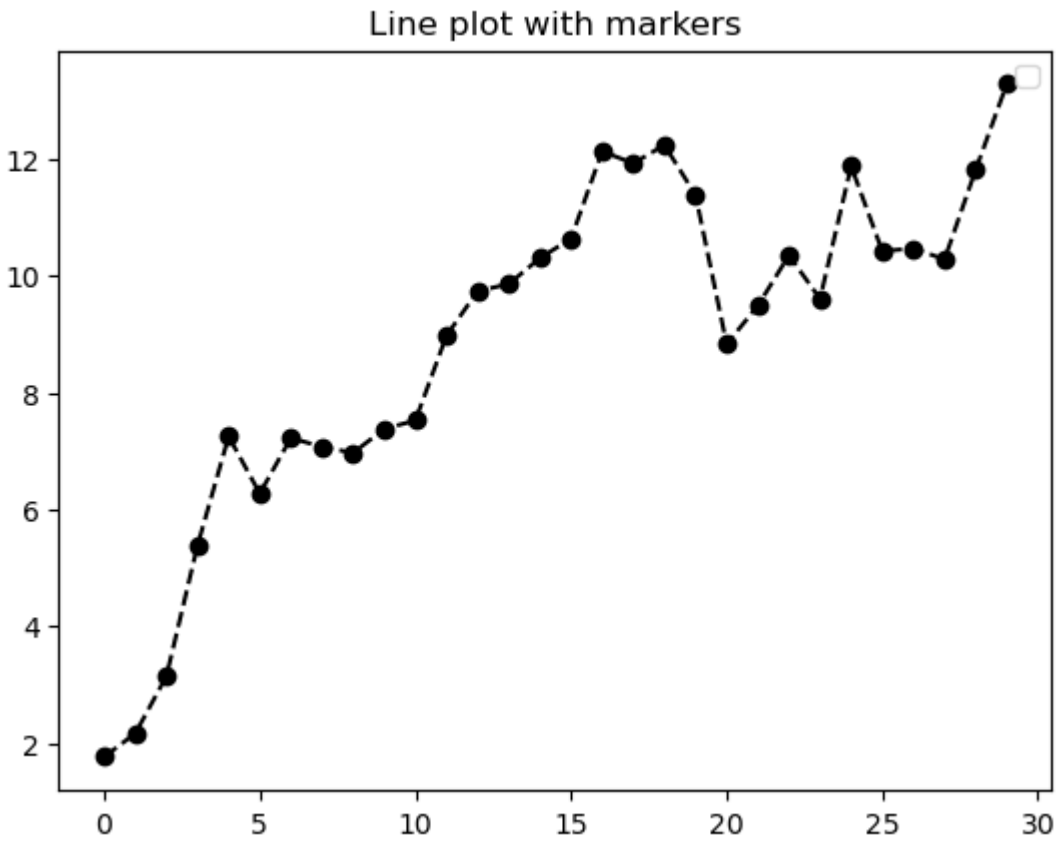
```
C:\Users\miran\AppData\Local\Temp\ipykernel_11480\2096456126.py:12: UserWarni
ng: color is redundantly defined by the 'color' keyword argument and the fmt
string "ko--" (-> color='k'). The keyword argument will take precedence.
  plt.plot(randn(30).cumsum(), 'ko--', color = 'g')              #color =
'g': 绿色
```

Out[10]:
```
Text(0.5, 1.0, 'Line plot with markers')
```

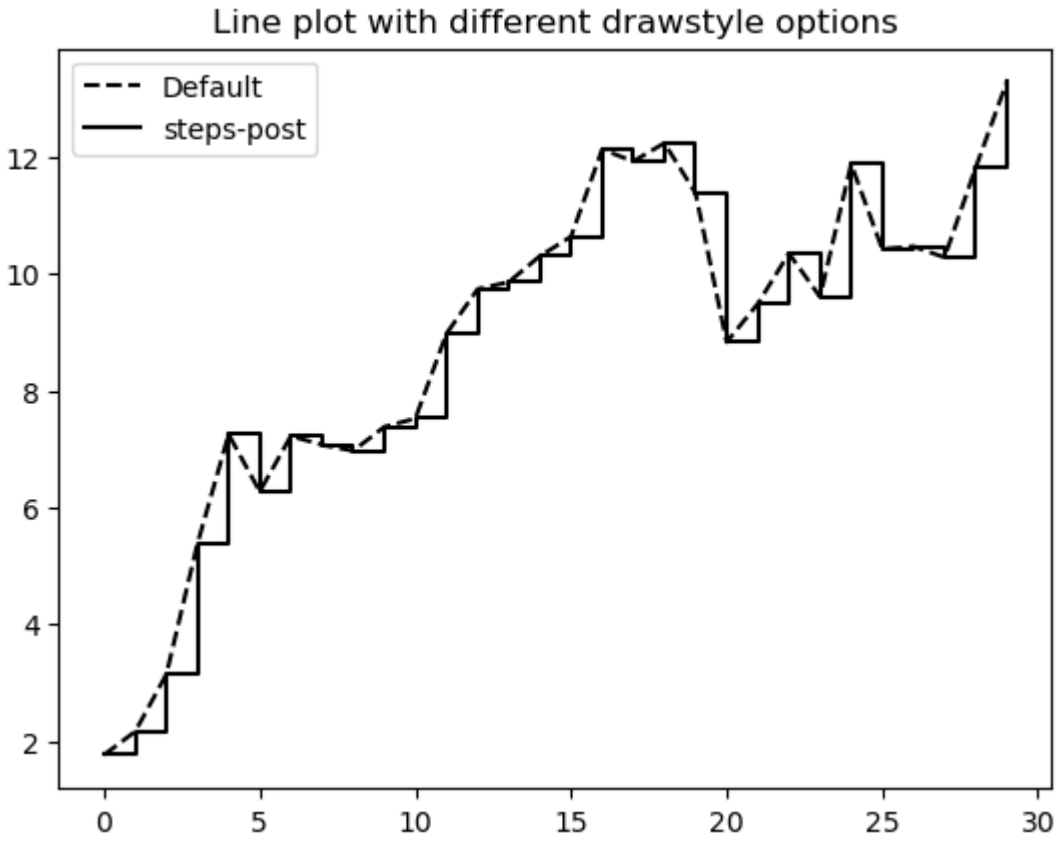


In [11]:
```
# #如果运行代码没有出图，可以在command输入：pip install numpy matplotlib
import numpy as np
import matplotlib.pyplot as plt

# 生成随机数据
np.random.seed(0)
data = np.random.randn(30).cumsum()

# 绘制图形
# color='k': 设置线条的颜色为黑色（k 是黑色的缩写）。
# linestyle='dashed': 设置线条的样式为虚线。
# marker='o': 设置数据点的标记样式为圆圈
plt.plot(data, color='k', linestyle='dashed', marker='o')            #黑色虚线
plt.title("Line plot with markers")
plt.legend(loc='best')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



In [12]:
```
#不同drawstyle选项下的折线图

# 生成随机数据
np.random.seed(0)
data = np.random.randn(30).cumsum()

# 绘制图形
plt.plot(data, 'k--', label='Default')                                    #黑虚
plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')          #黑实
plt.legend(loc='best')
plt.title("Line plot with different drawstyle options")
plt.show()
#legend 函数用于创建图例，参数 loc='best' 表示图例位置的最佳选择。根据图形的布局和内
#plt.show()
```



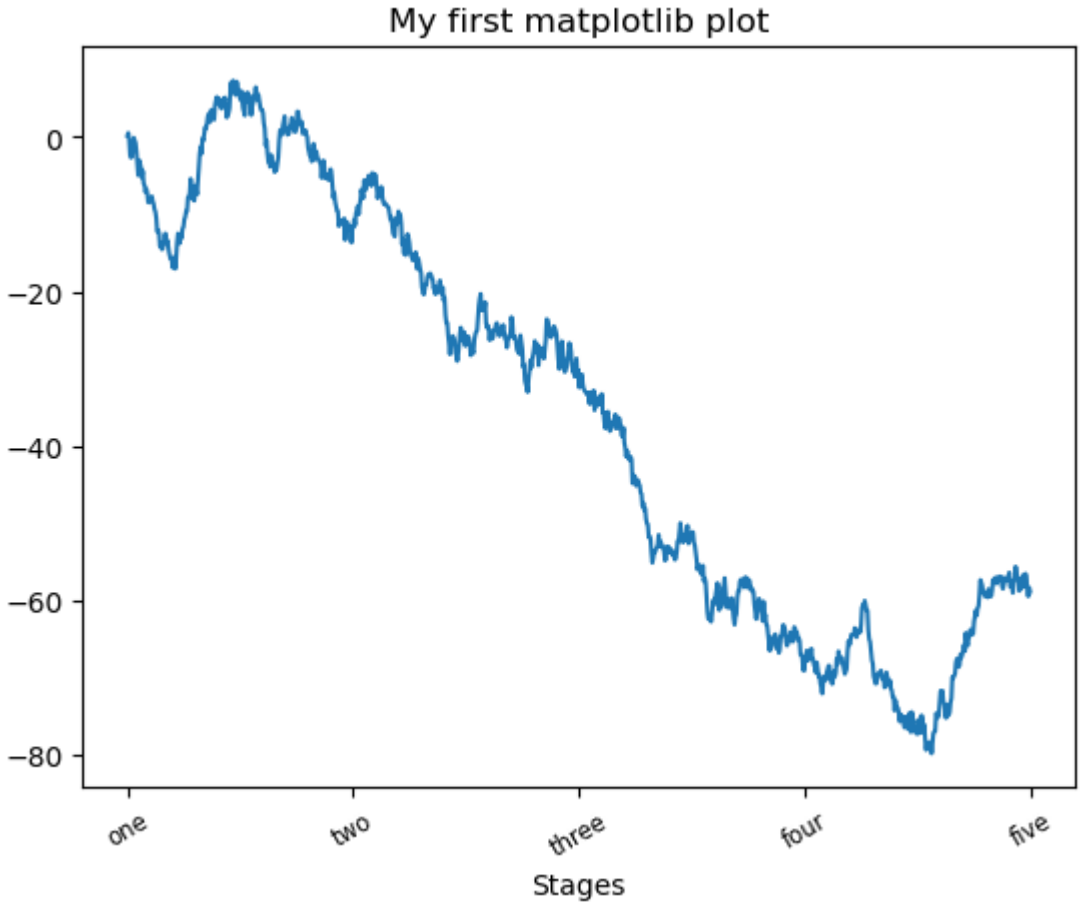
9.1.3 刻度、标签和图例 Ticks, Labels, and Legends

In [13]:
```
#大多数图标修饰，使用程序性pyplot接口；和更多面向对象的原生matplotlib API

#pyplot接口设计为交互式使用，包含了xlim, xticks, xticklabels。分别控制了绘图范围，
#在没有函数参数下调用，返回当前参数值，plt.xlim()返回当前的x轴绘图范围
#传入参数调用，并设置参数值：plt.xlim([0, 10])会将x轴范围设置为0到10
```

9.1.3.1 设置标题、轴标签、刻度和刻度标签

In [14]:
```
#绘制随机漫步：
#改变x轴刻度，最简单的方式是使用set_xticks和set_yticklabels。
#set_xticks表示在数据范围内设定刻度的位置，为标签赋值。
#ratotation将x轴刻度标签旋转30度。
#set_xlable给x轴一个名称，set_title给子图标题。


fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(np.random.randn(1000).cumsum())

#set_xticks 是用于设置 x 轴刻度位置的方法，参数 [0, 250, 500, 750, 1000] 表示希望
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                            rotation = 30, fontsize = 'small')
ax.set_title('My first matplotlib plot')
ax.set_xlabel('Stages')
```

Out[14]: Text(0.5, 0, 'Stages')

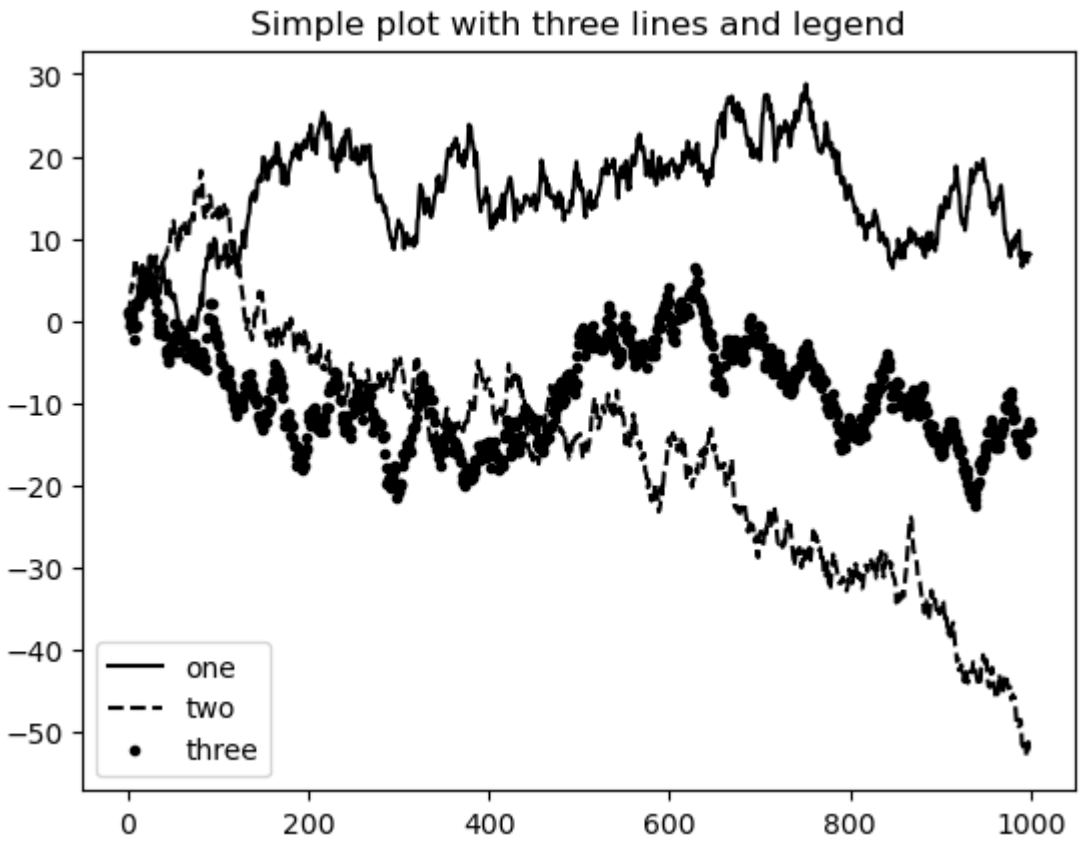


In [15]:
```
#修改y轴坐标是相同过程，将上面示例中的x替换成y即可。轴的类型拥有一个set方法，允许批量
#在调用 ax.set(**props) 之前，你需要先创建一个图形对象和坐标轴对象，并将其赋值给相应
#然后，使用这个坐标轴对象来操作和设置图形的属性。
#props 字典包含了两个键值对：'title' 和 'xlabel'。
#'title' 键对应的值是字符串 'My first matplotlib plot'，表示设置图形的标题为该字符

# 设置标题和 x 轴标签
props = {
    'title': 'My first matplotlib plot',
    'xlabel': 'Stages'
}

# 创建图形和坐标轴对象
fig, ax = plt.subplots()

# 设置标题和 x 轴标签
ax.set(**props)

# 添加绘图代码
# 例如：绘制随机数据的折线图
data = [1, 2, 3, 4, 5]
plt.plot(data, 'k-')

# 显示图形
plt.show()
```

9.1.3.2 添加图例

In [16]:
```
#图例是用于区分绘图元素的另一个重要内容。最简单的方式是在添加每个图表时，传递label参数
from numpy.random import randn
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'k', label = 'one')                #黑实
ax.plot(randn(1000).cumsum(), 'k--', label = 'two')              #黑虚
ax.plot(randn(1000).cumsum(), 'k.', label = 'three')             #黑点
ax.legend(loc = 'best')                                          #图例放在最佳位置
ax.set_title('Simple plot with three lines and legend')
```

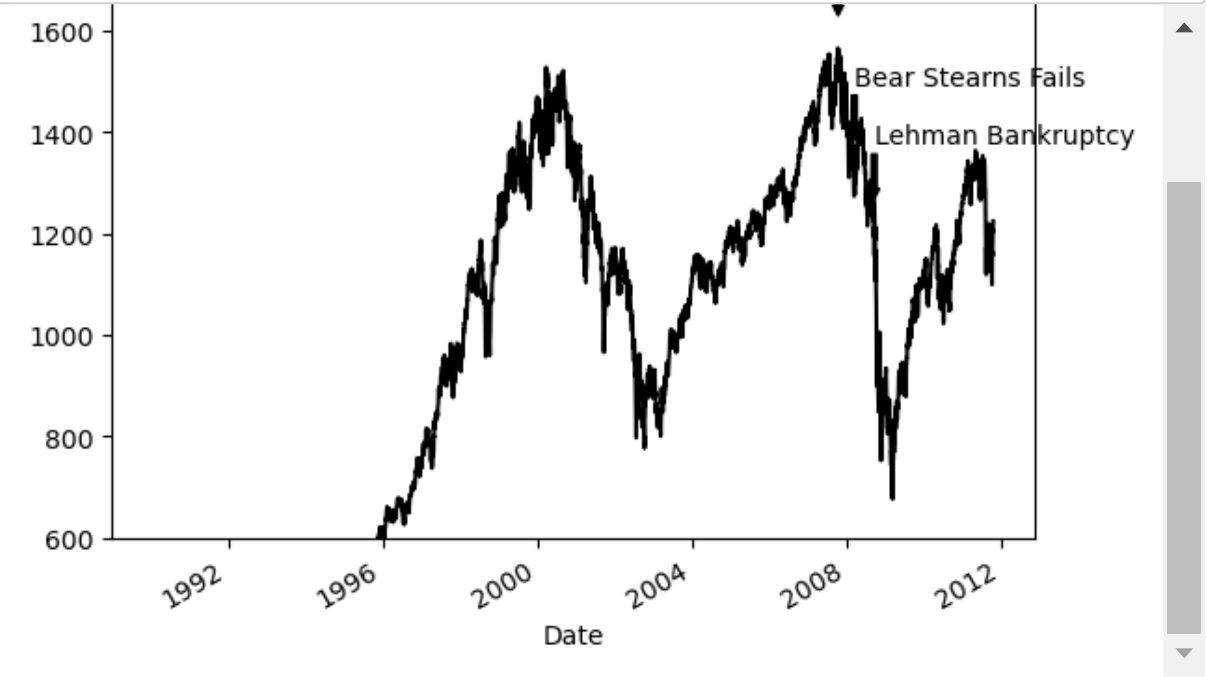
Out[16]:
```
Text(0.5, 1.0, 'Simple plot with three lines and legend')
```



9.1.4 注释与子图加工 Adding legends& Annotations and Drawing on a Subplot

In [17]:
```
#可能会想再图标上绘制注释，注释中可能包含文本，箭头，其他图形。可以使用text, arrow,
#text在图标上给定的坐标（x, y），根据可选的定制样式绘制文本。

# ax.text(x, y, 'Hello World!',
#         family = 'monospace', fontsize = 10)
```

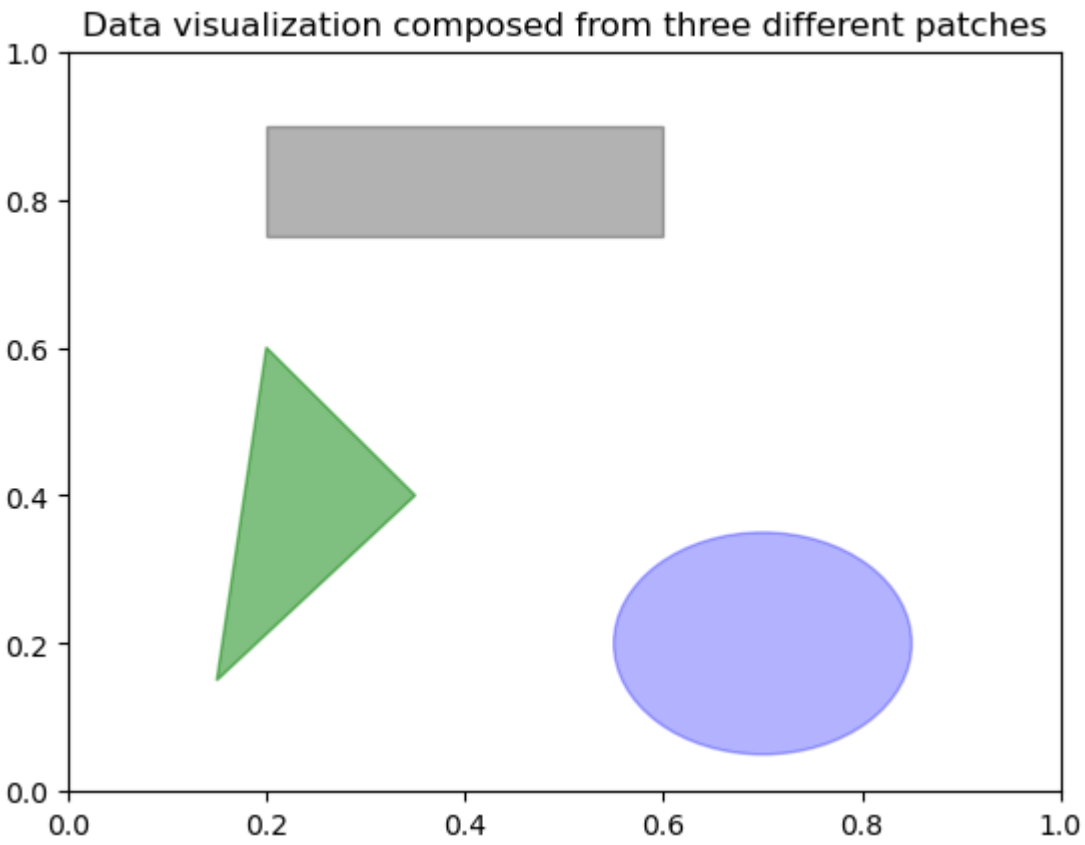
In [18]:
```
#注释可以同时绘制文本和箭头。
#绘制标普500指数从2007年以来的收盘价。

from datetime import datetime

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

data = pd.read_csv('C:/Users/miran/lpthw/spx.csv', index_col = 0, parse_dates
spx = data['SPX']
spx.plot(ax = ax, style = 'k-')

crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]

for date, label in crisis_data:
    ax.annotate(label, xy = (date, spx.asof(date) + 75),
                xytext = (date, spx.asof(date) + 225),
                    arrowprops = dict(facecolor = 'black', headwidth = 4, widt
                horizontalalignment = 'left', verticalalignment = 'top')

#可以set_xlim和set_ylim手动设置图表边界。放大时间横轴和纵轴。
#ax.set_xlim(['1/1/2007', '1/1/2011'])
ax.set_ylim([600, 1800])
ax.set_title('Important dates in the 2008-2009 financial crisis')
```



In [19]:
```
#matplotlib含有多种常见图形对象，对象的引用是patches。
#一些图形如rectangle矩形和circle圆形，可以在matplotlib.pyplot中找到，但图形全集位于
```

In [20]:
```
#想在图表中添加图形时，需要生成patch补丁对象shp，调用ax.add_patch(shp)将它加入到子图

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

rect = plt.Rectangle((0.2, 0.75), 0.4, 0.15, color = 'k', alpha = 0.3)        #
circ = plt.Circle((0.7, 0.2), 0.15, color = 'b', alpha = 0.3)                 #
pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]],                   #
                   color = 'g', alpha = 0.5)

ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
ax.set_title('Data visualization composed from three different patches')
```

Out[20]:
```
Text(0.5, 1.0, 'Data visualization composed from three different patches')
```



9.1.5 将图片保存到文件 Saving Plots to File

In [21]:
```
#可以使用plt.savefig将活动图片保存到文件。
plt.savefig('figpath.svg')
plt.savefig('figpath.png', dpi = 400, bbox_inches = 'tight')
#save
```

```
<Figure size 640x480 with 0 Axes>
```

9.1.6 matplotlip设置 matplotlib Configuration

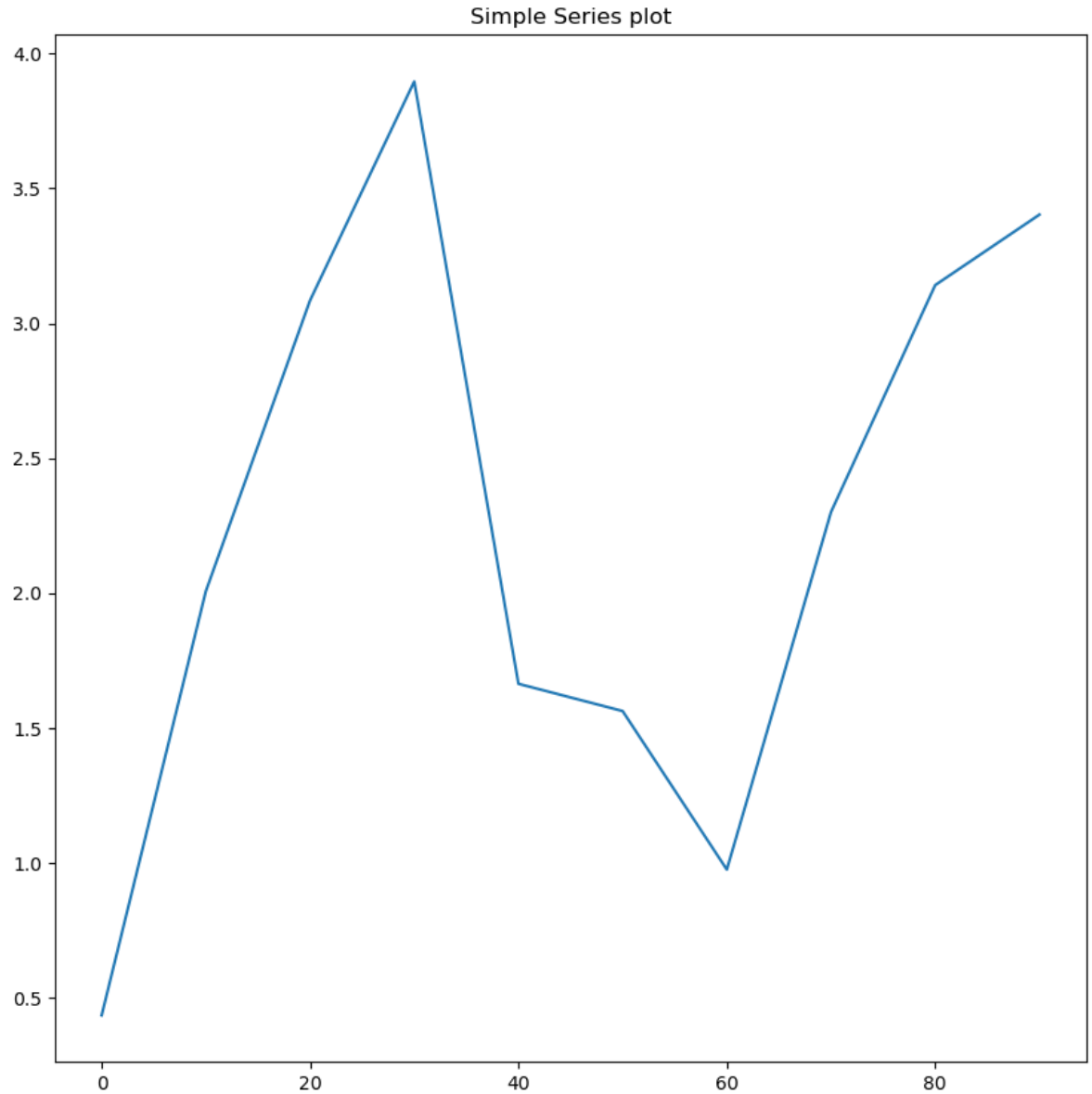
In [22]:
```
#将全局默认数字大小设置为10*10,
plt.rc('figure', figsize = (10, 10))
```

9.2 使用Pandas和seaborn绘图 Plotting with pandas and seaborn

9.2.1 折线图 Line Plots

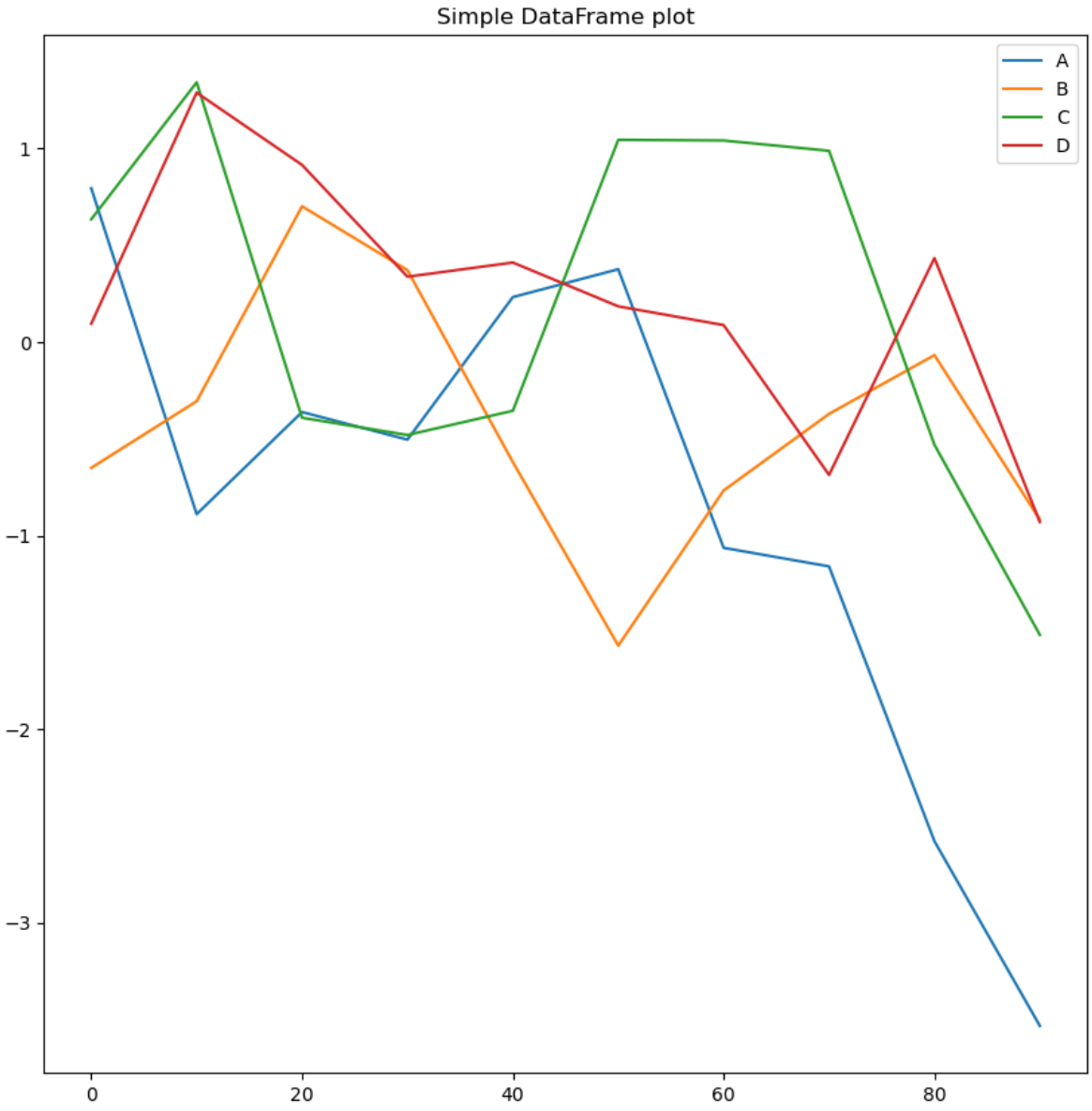
In [23]:
```
s = pd.Series(np.random.randn(10).cumsum(), index = np.arange(0, 100, 10))
s.plot()
plt.title("Simple Series plot")
```

Out[23]:
```
Text(0.5, 1.0, 'Simple Series plot')
```



In [24]:
```
#Series对象的索引传入matplotlib作为绘图的x轴，可以通过传入use_index = False禁用功能
#x轴刻度和范围，通过xticks和xlim调整，y轴通过yticks和ylim调整。
#大部分pandas的绘图方法，接收可选的ax参数，该参数可以是一个matplotlib子图对象。这使

df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                  columns = ['A', 'B', 'C', 'D'],
                  index = np.arange(0, 100, 10))
df
df.plot()
plt.title("Simple DataFrame plot")
#df.plot.line()
#df.plot等价于df.plot.line()


# np.random.randn(10, 4)生成了一个形状为(10, 4)的随机数数组，表示10行4列的数据。
# 然后，通过cumsum(0)对每一列进行累积求和操作，生成了每一列的累积和。这样得到的结果被

# columns参数指定了DataFrame的列标签，分别为'A'、'B'、'C'和'D'。
# index参数指定了DataFrame的行索引，使用np.arange(0, 100, 10)生成了从0到90的数列作

# 最终，得到的df对象表示一个包含了累积和的数据框，具有'A'、'B'、'C'和'D'四列，以及0到
```

Out[24]:
```
Text(0.5, 1.0, 'Simple DataFrame plot')
```



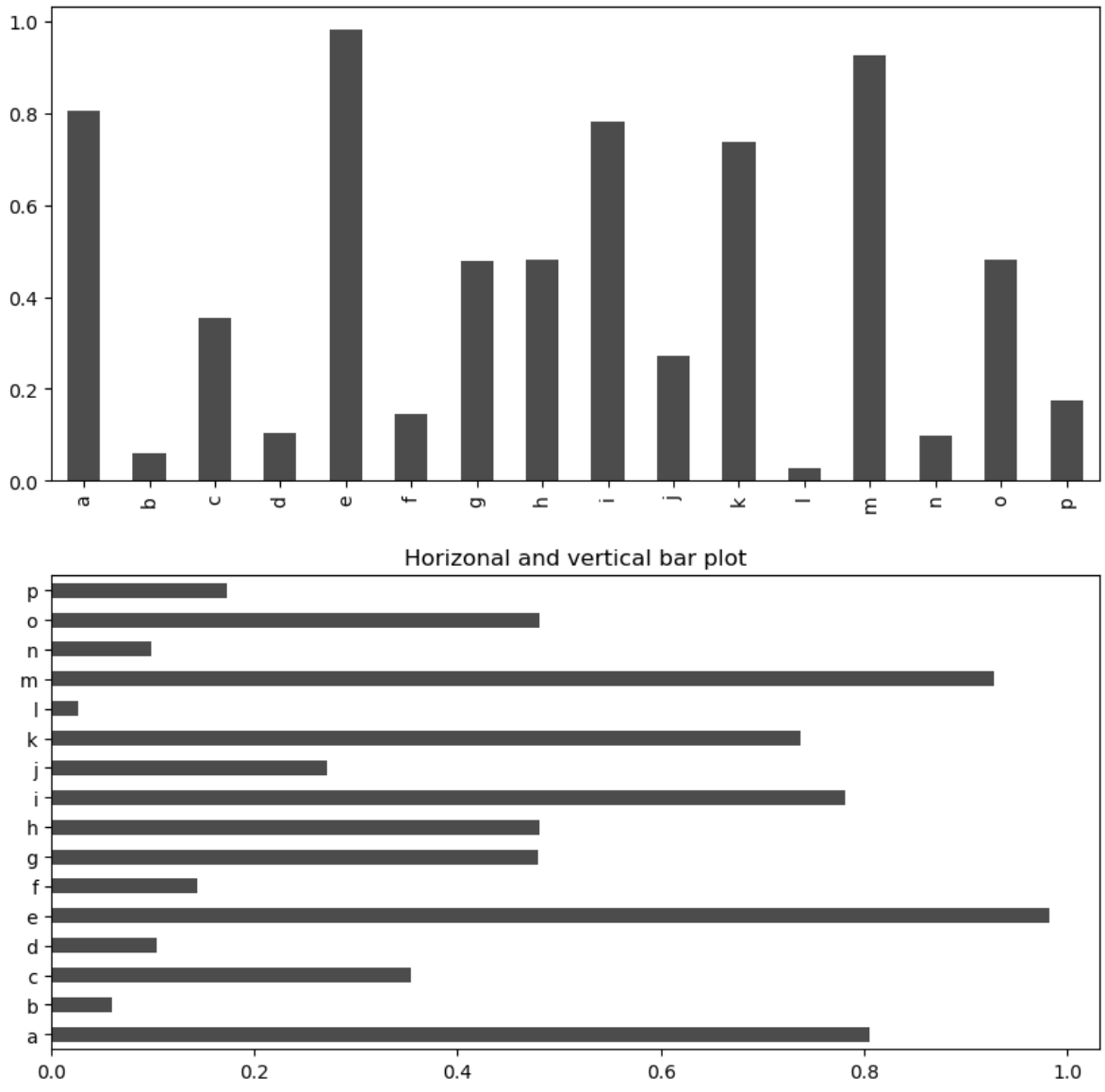
In [25]:
```
#series.plot方法参数

#label：图例标签
#ax：绘图所用的matplotlib子图对象，如果没传值，则使用当前活动的matplotlib子图
#style：传给matplotlib的样式字符串，比如'ko--'
#alpha：图片不透明度(0-1)
#kind：可以是area, bar, barth, density, hist, kde, line, pie
#logy：在y轴上使用对数缩放
#use_index：使用对象索引刻度标签
#rot：刻度标签的旋转(0到360)
#xticks：用于x轴刻度的值
#yticks：用于y轴刻度的值
#xlim：x轴范围(eg. [0, 10])
#ylim：y轴范围
#grd：展示轴网格
```

In [26]:
```
#data frame的plot参数
#subplots：将data frame每一列绘制在独立的子图中
#sharex：如果subplots = True，则共享相同的x轴，刻度和范围
#sharey：如果subplots = True，则共享相同的y轴
#figsize：用于生成图片尺寸的元组
#title：标题字符串
#lengend：添加子图图例
#sort_columns：按字母顺序绘制各列，默认情况下使用已有的列顺序
```

9.2.2 柱状图 Bar Plots

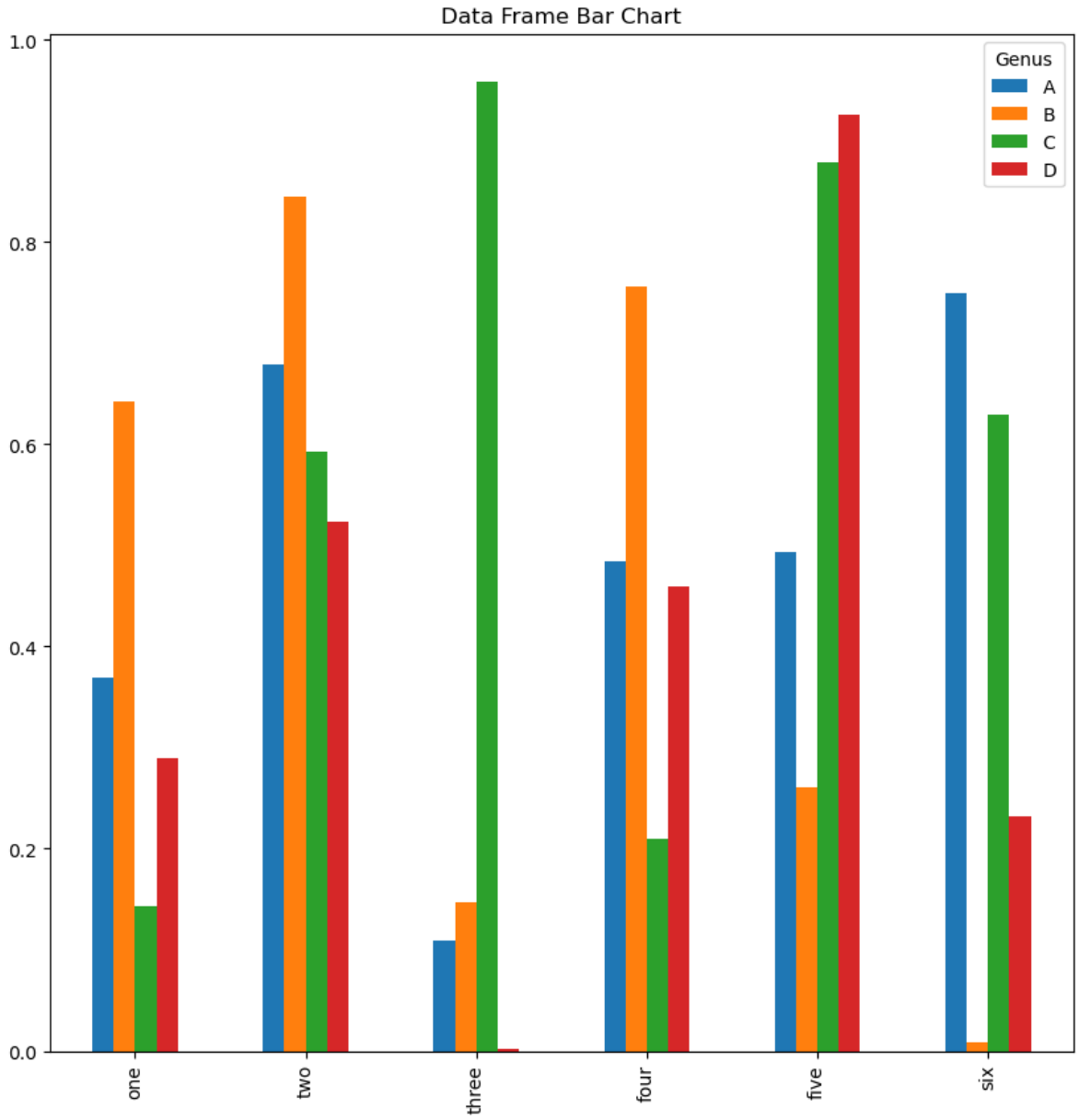
In [27]:
```
#plot.bar()和plot.barh()可以分别绘制垂直和水平的柱状图。
#绘制柱状图时，series或Data frame的索引将会被用作x轴刻度(bar)或y轴刻度(barh)

#Series绘图:
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index = list('abcdefghijklmnop'))
data

data.plot.bar(ax = axes[0], color = 'k', alpha = 0.7)
# ax = axes[0]指定了绘图的目标子图，即将柱状图绘制在axes中的第一个子图axes[0]上。
#'k'代表黑色，'r'代表红色，'b'代表蓝色，'g'代表绿色等
# alpha = 0.7指定了柱状图的透明度，这里设置为0.7，表示相对较不透明

data.plot.barh(ax = axes[1], color = 'k', alpha = 0.7)
plt.title("Horizonal and vertical bar plot")
```

Out[27]:
```
Text(0.5, 1.0, 'Horizonal and vertical bar plot')
```



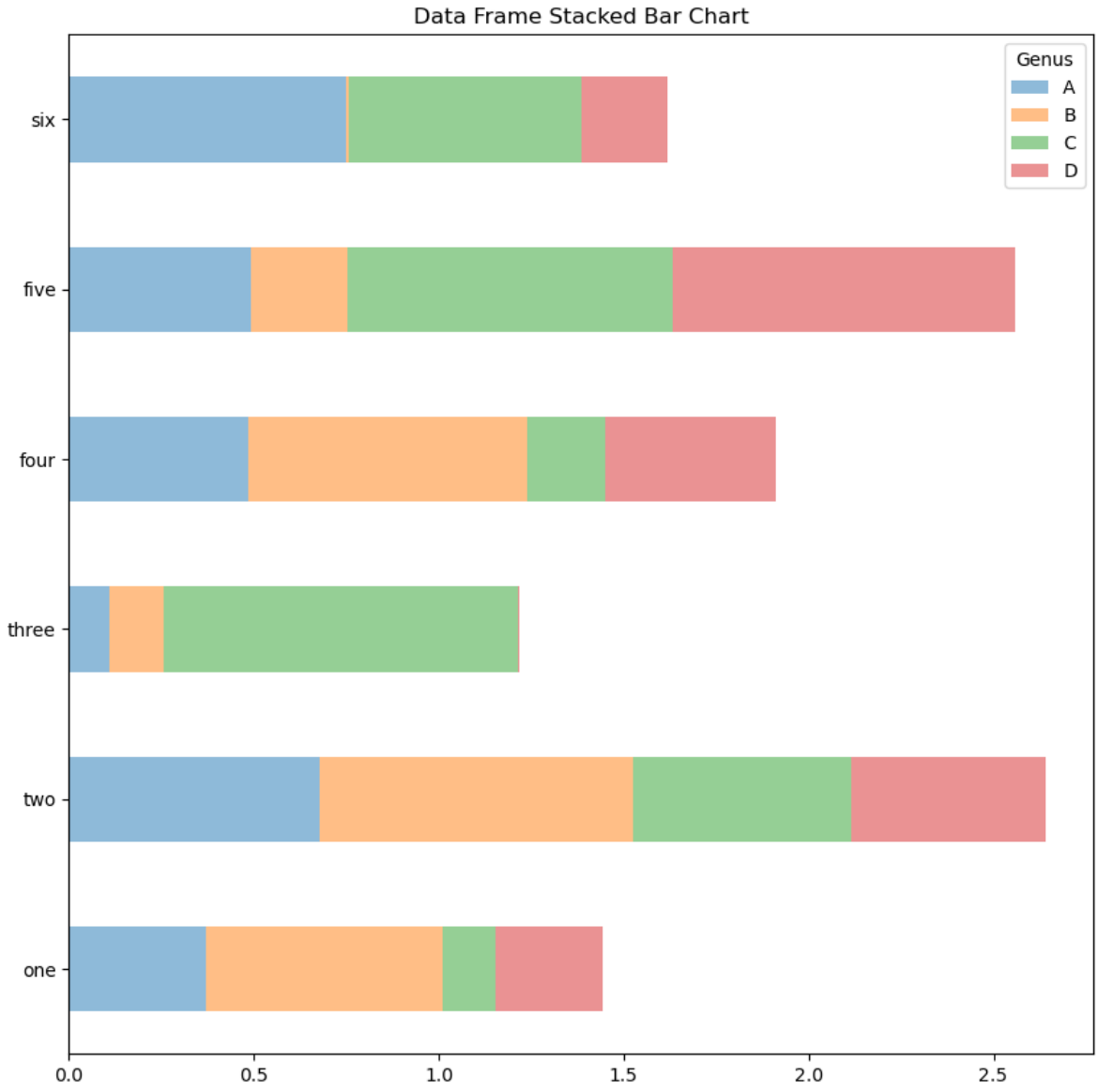
In [28]:
```
#data frame中，柱状图将每一行中的值，分组到并排的柱子中的一组。
df = pd.DataFrame(np.random.rand(6, 4),
                  index = ['one', 'two', 'three', 'four', 'five', 'six'],
                  columns = pd.Index(['A', 'B', 'C', 'D'], name = 'Genus'))
df
# Genus A   B   C   D
# one   0.823718    0.232773    0.310629    0.791227
# two   0.715143    0.558051    0.704948    0.418637
# three 0.005310    0.011355    0.511222    0.083291
# four  0.051075    0.965517    0.859003    0.152027
# five  0.000664    0.941668    0.278325    0.185898
# six   0.691508    0.108904    0.264650    0.975095

df.plot.bar()
plt.title("Data Frame Bar Chart")
```

Out[28]:
```
Text(0.5, 1.0, 'Data Frame Bar Chart')
```



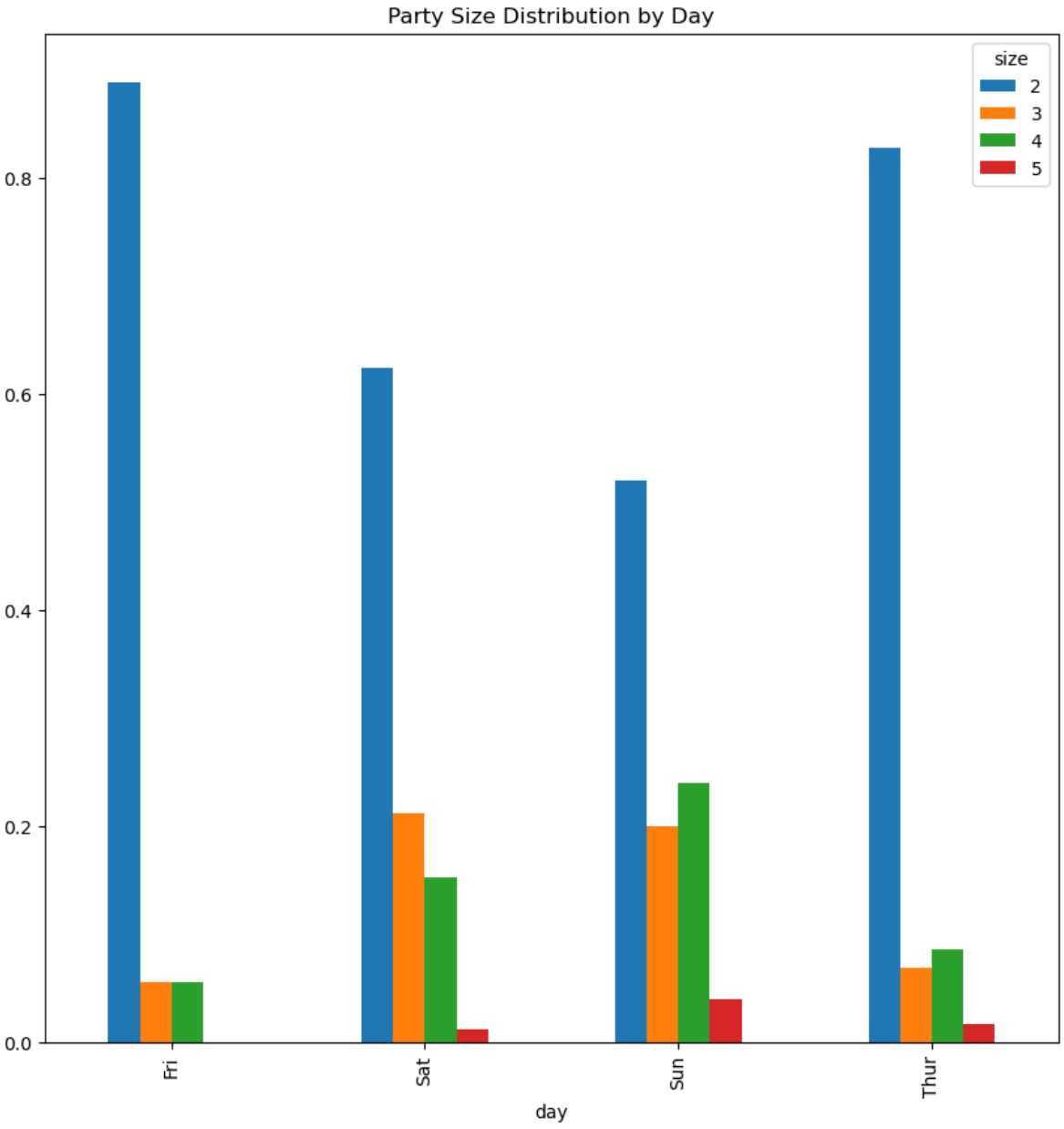
In [29]:
```
# 注意data frame的列名称 genus被用作了图例标题。
# 可以通过传递stacked = True 来生成堆积柱状图，使得每一行的值堆积一起。
df.plot.barh(stacked = True, alpha = 0.5)
plt.title("Data Frame Stacked Bar Chart")
```

Out[29]: Text(0.5, 1.0, 'Data Frame Stacked Bar Chart')



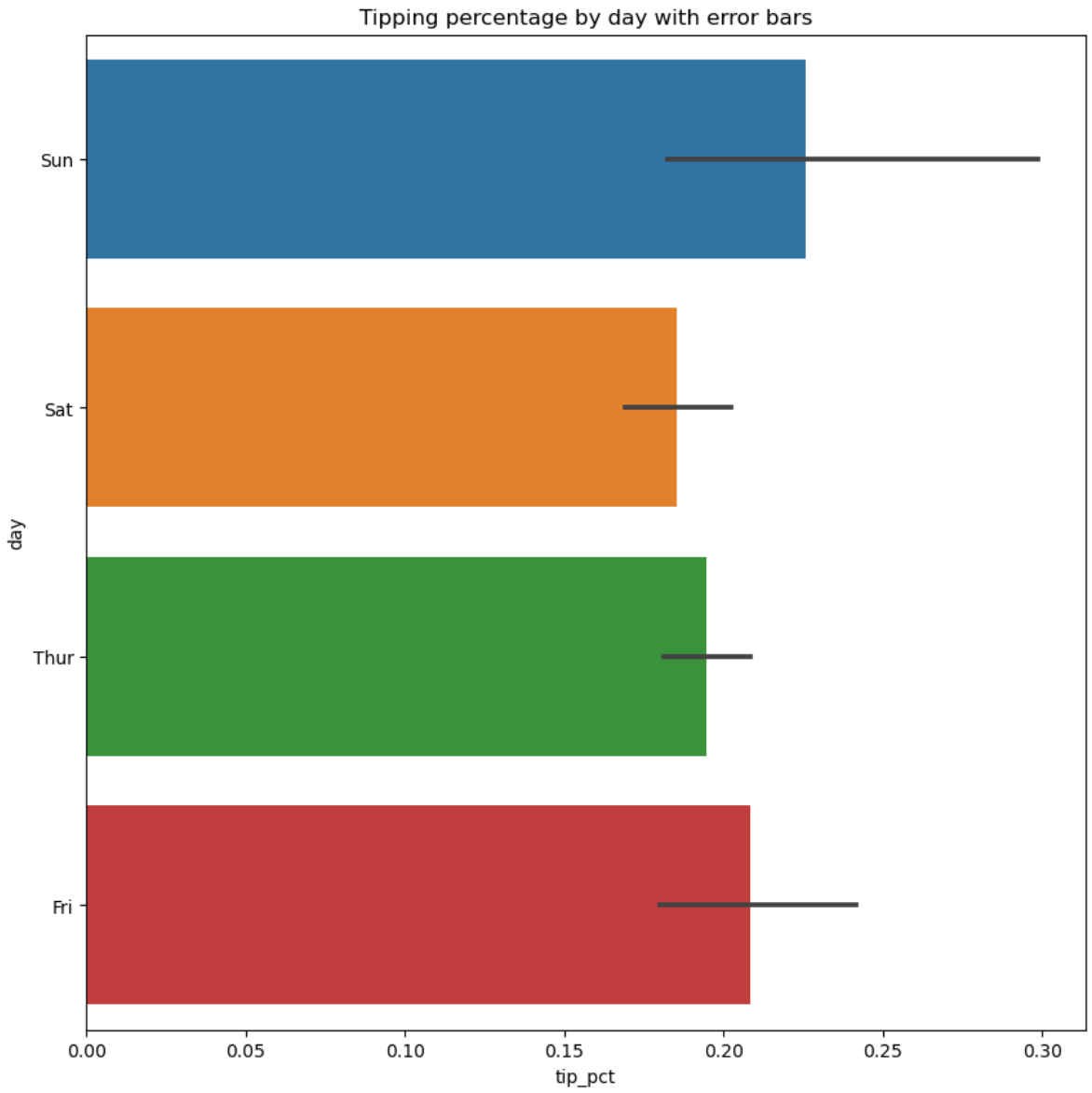
In [30]:
```
#使用value_counts: s.value_counts().plot.bar()可以有效对series值频率进行可视化。

tips = pd.read_csv('C:/Users/miran/lpthw/tips.csv')
tips

#使用tips数据框中的两列'day'和'size'作为参数，使用pd.crosstab()函数创建一个交叉表，
#这将生成一个新的数据框party_counts，它显示了每个星期几（'day'）和不同聚会大小（'si
party_counts = pd.crosstab(tips['day'], tips['size'])
party_counts
# 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

#没有太多人参加1,6人排队，所以只保留2-5列
party_counts = party_counts.loc[:, 2:5]


#下面计算了每个星期几（'day'）和聚会大小（'size'）组合的百分比。
#party_counts.sum(1)计算了每行的总和，即每个星期几的总计数。
#party_counts.div(party_counts.sum(1), axis=0)使用div()函数将party_counts中的每
#结果生成一个新的数据框party_pcts，其中包含了每个星期几和聚会大小组合的百分比。每个值
#进行标准化以确保每一行的值和为1，绘图:
party_pcts = party_counts.div(party_counts.sum(1), axis = 0)
party_pcts
# 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
party_pcts.plot.bar()
plt.title("Party Size Distribution by Day")

#可以看到本数据集中的排队数量在周末增加。
```

Out[30]:
```
Text(0.5, 1.0, 'Party Size Distribution by Day')
```


Party Size Distribution by Day
size
2
3
4
5
0.0
0.2
0.4
0.6
0.8
Fri
Sat
Sun
Thur
day

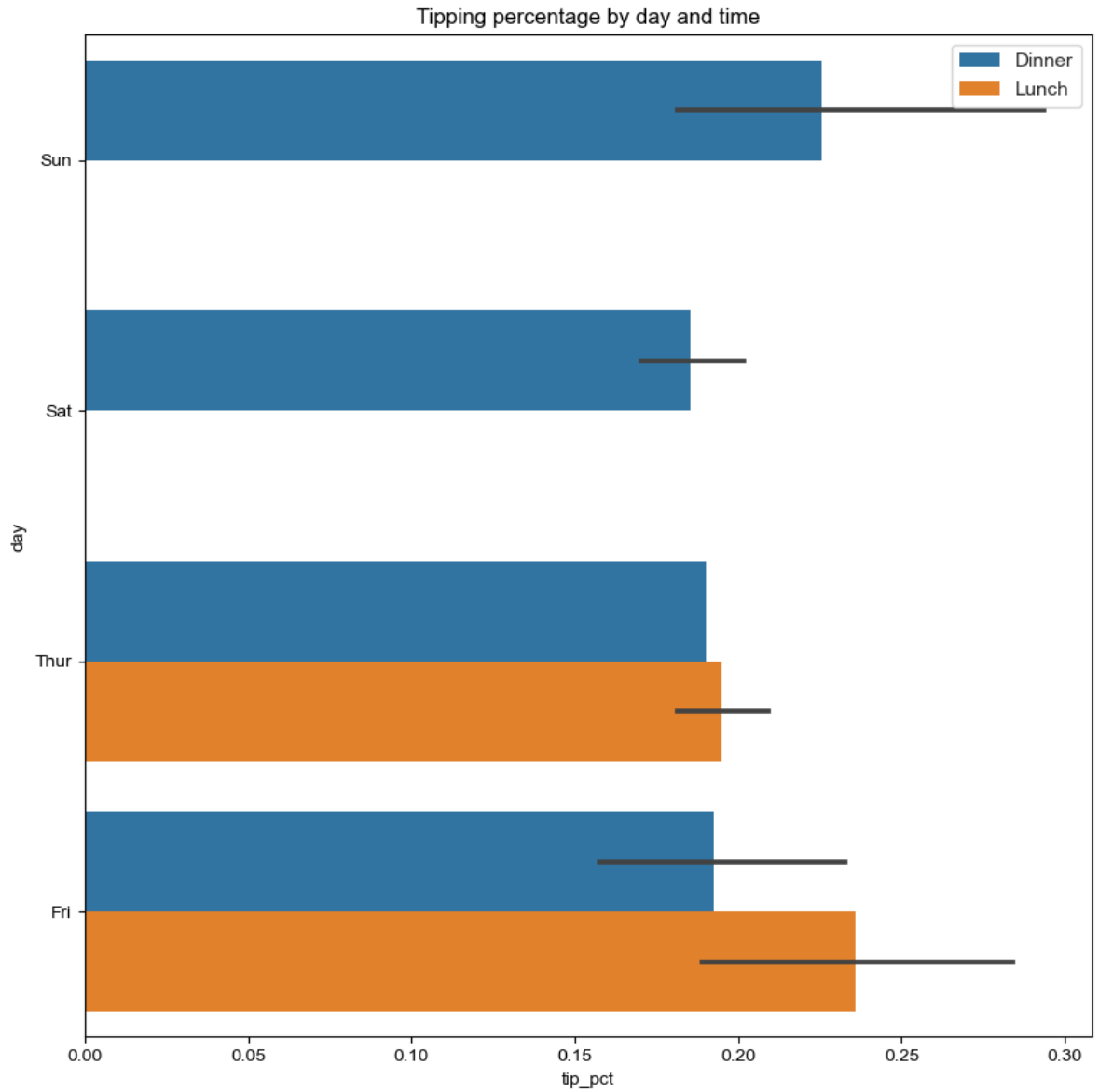
In [31]:
```
#在绘图前需要聚合汇总的数据，使用Seaborn包会使工作更为简单。现在看下使用seaborn进行

import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])     #创建一
tips.head()
sns.barplot(x = 'tip_pct', y = 'day', data = tips, orient = 'h')
plt.title("Tipping percentage by day with error bars")

# 该代码使用Seaborn库中的barplot函数绘制水平柱状图。
# 它通过指定x和y参数来确定要绘制的数据，即'tip_pct'和'day'。
# data参数指定了要使用的数据集，即变量名为tips的DataFrame。orient参数设置为'h'，表
```

Out[31]: Text(0.5, 1.0, 'Tipping percentage by day with error bars')



In [32]:
```
#seaborn中的绘图函数使用一个Data参数，这个参数可以是pandas的data frame。其他的参数
#seaborn.barplot有一个hue选项，允许通过一个额外的分类值，将数据分离：
#因为day列中有多个观测值，柱子的值是tip_pct的平均值。
#柱子上的黑线，代表95%置信区间。

#使用Seaborn绘制柱状图，x轴表示'tip_pct'，y轴表示'day'，通过'hue'参数指定'time'列
#'orient'参数设置为'v'，表示绘制垂直柱状图。
sns.barplot(x = 'tip_pct', y = 'day', hue ='time', data = tips, orient  = 'h')
sns.set(style = 'whitegrid')
plt.legend(loc='best')
plt.title("Tipping percentage by day and time")
```

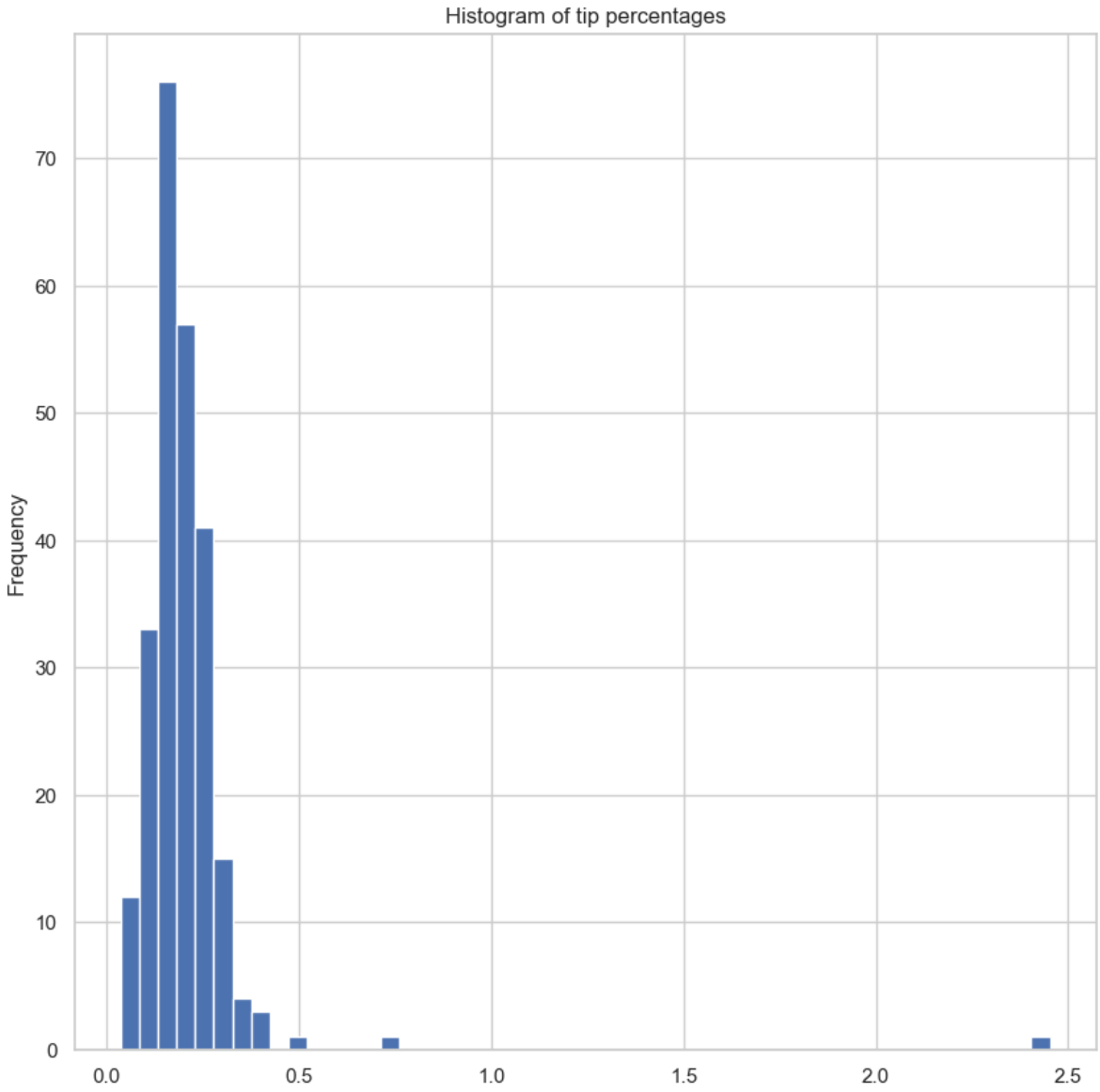
Out[32]:
```
Text(0.5, 1.0, 'Tipping percentage by day and time')
```



9.2.3 直方图和密度图 Histograms and Density Plots

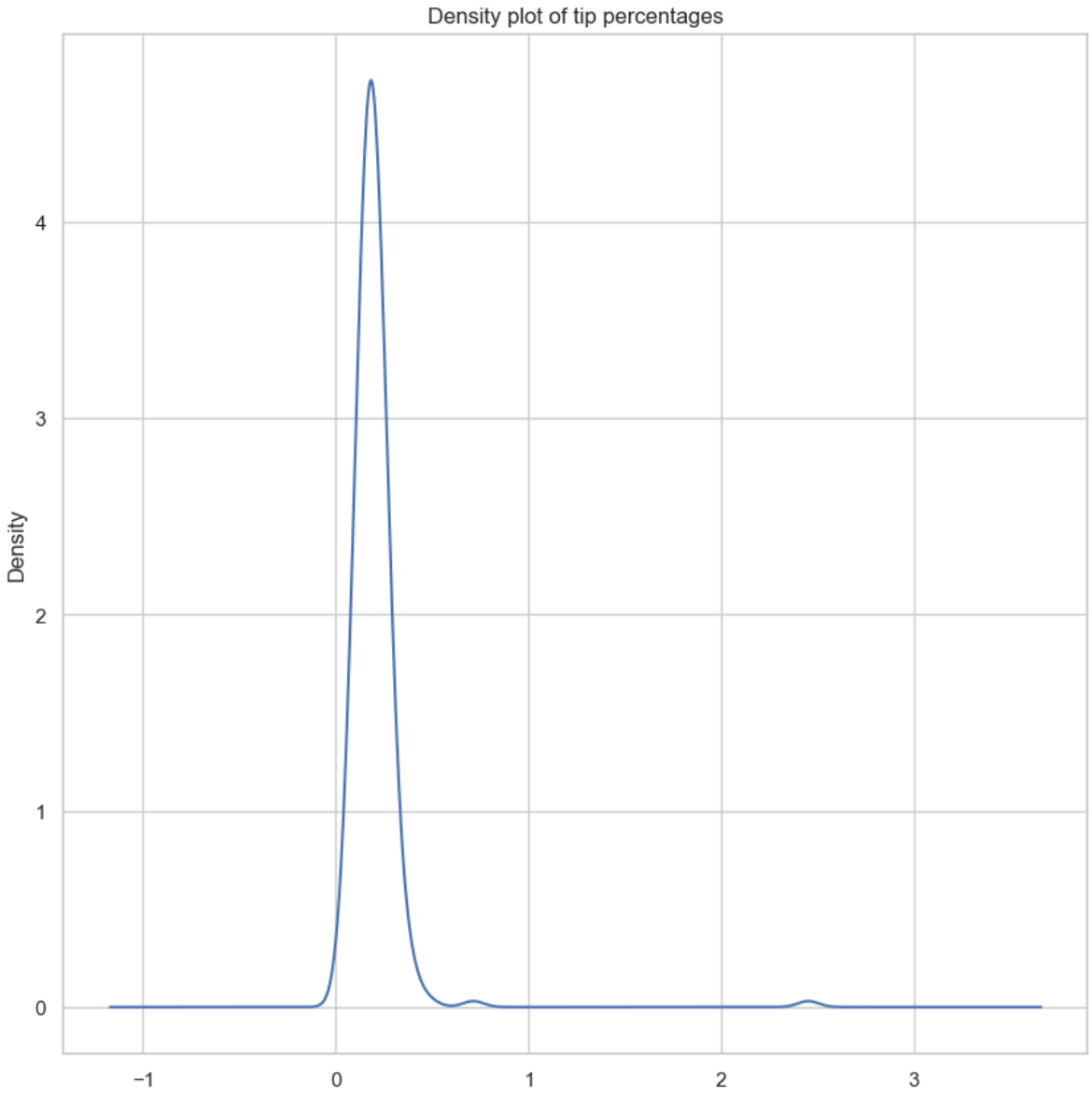
In [33]:
```
#直方图是一种条形图，用于给出值频率的离散显示。
#数据点被分成离散的，均匀间隔的箱，并且绘制每个箱中数据点的数量。
#使用之前的小费数据，可以使用series的plot.hist方法制作小费站总费用百分比的直方图。


tips['tip_pct'].plot.hist(bins = 50)
plt.title("Histogram of tip percentages")
```

Out[33]:
```
Text(0.5, 1.0, 'Histogram of tip percentages')
```



In [34]:
```
tips['tip_pct'].plot.density()
plt.title("Density plot of tip percentages")
```

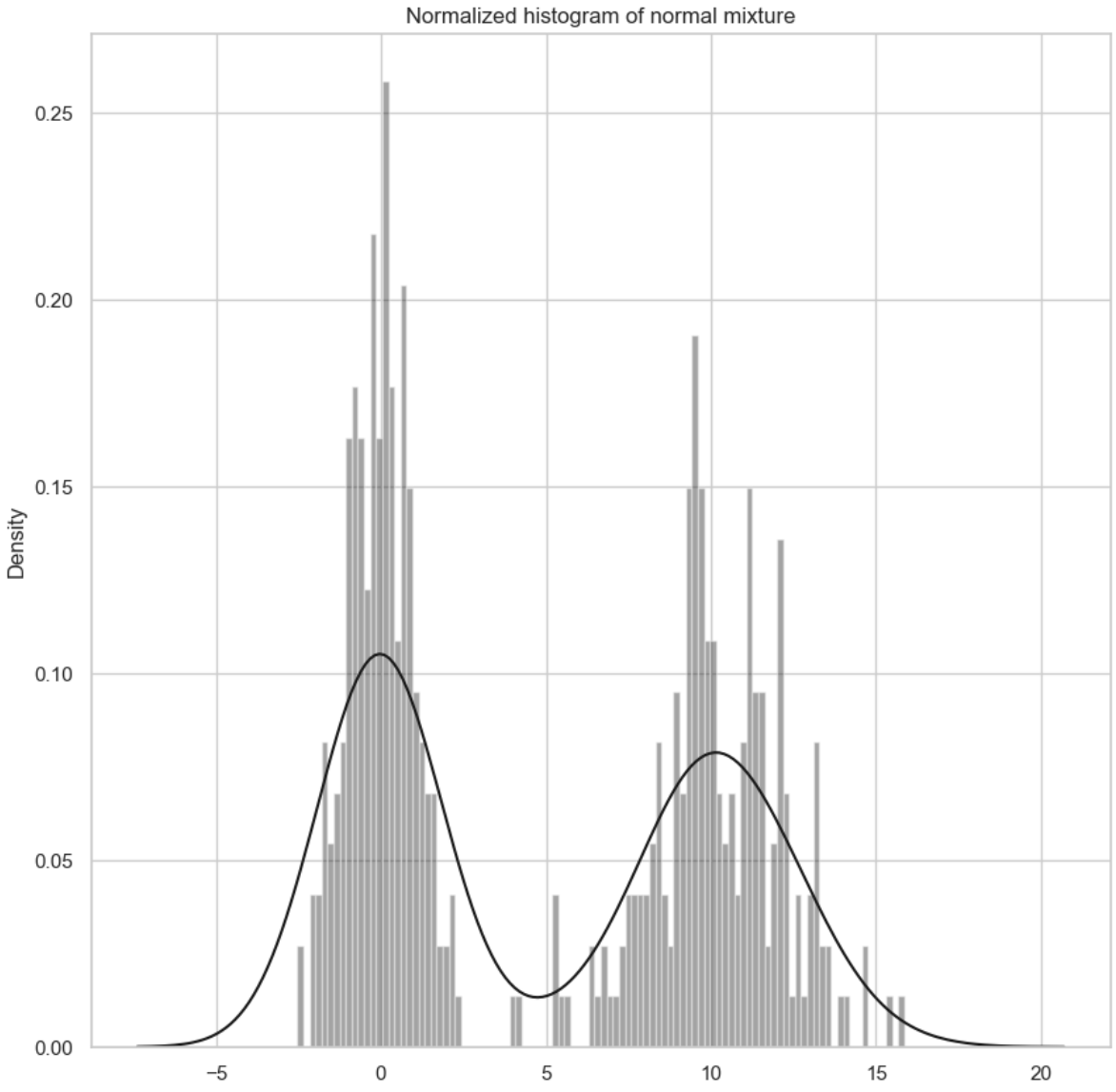
Out[34]:
```
Text(0.5, 1.0, 'Density plot of tip percentages')
```



In [35]:
```
#生成一个包含200个服从均值为0、标准差为1的正态分布随机数的数组
#生成一个包含200个服从均值为10、标准差为2的正态分布随机数的数组
#将comp1和comp2两个数组连接起来，生成一个包含所有元素的Series对象，并将其存储在名为v

comp1 = np.random.normal(0, 1, size = 200)
comp2 = np.random.normal(10, 2, size = 200)
values = pd.Series(np.concatenate([comp1, comp2]))
sns.distplot(values, bins = 100, color = 'k')
plt.title("Normalized histogram of normal mixture")
```

```
C:\Users\miran\AppData\Local\Temp\ipykernel_11480\2084729897.py:8: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(values, bins = 100, color = 'k')
```

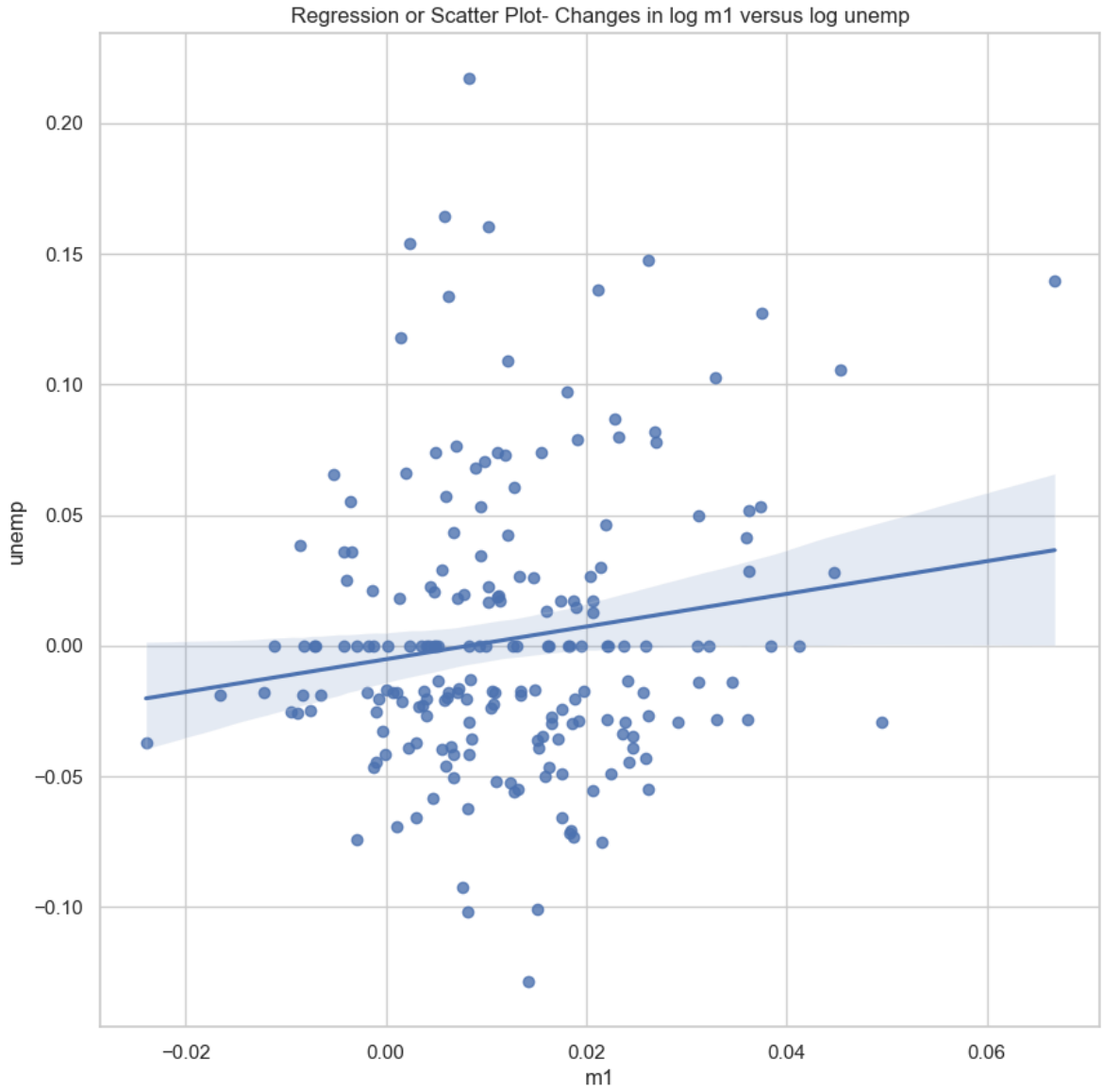
Out[35]:
```
Text(0.5, 1.0, 'Normalized histogram of normal mixture')
```


Normalized histogram of normal mixture
Density
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
20

9.2.4 散点图或点图 Scatter or Point Plots

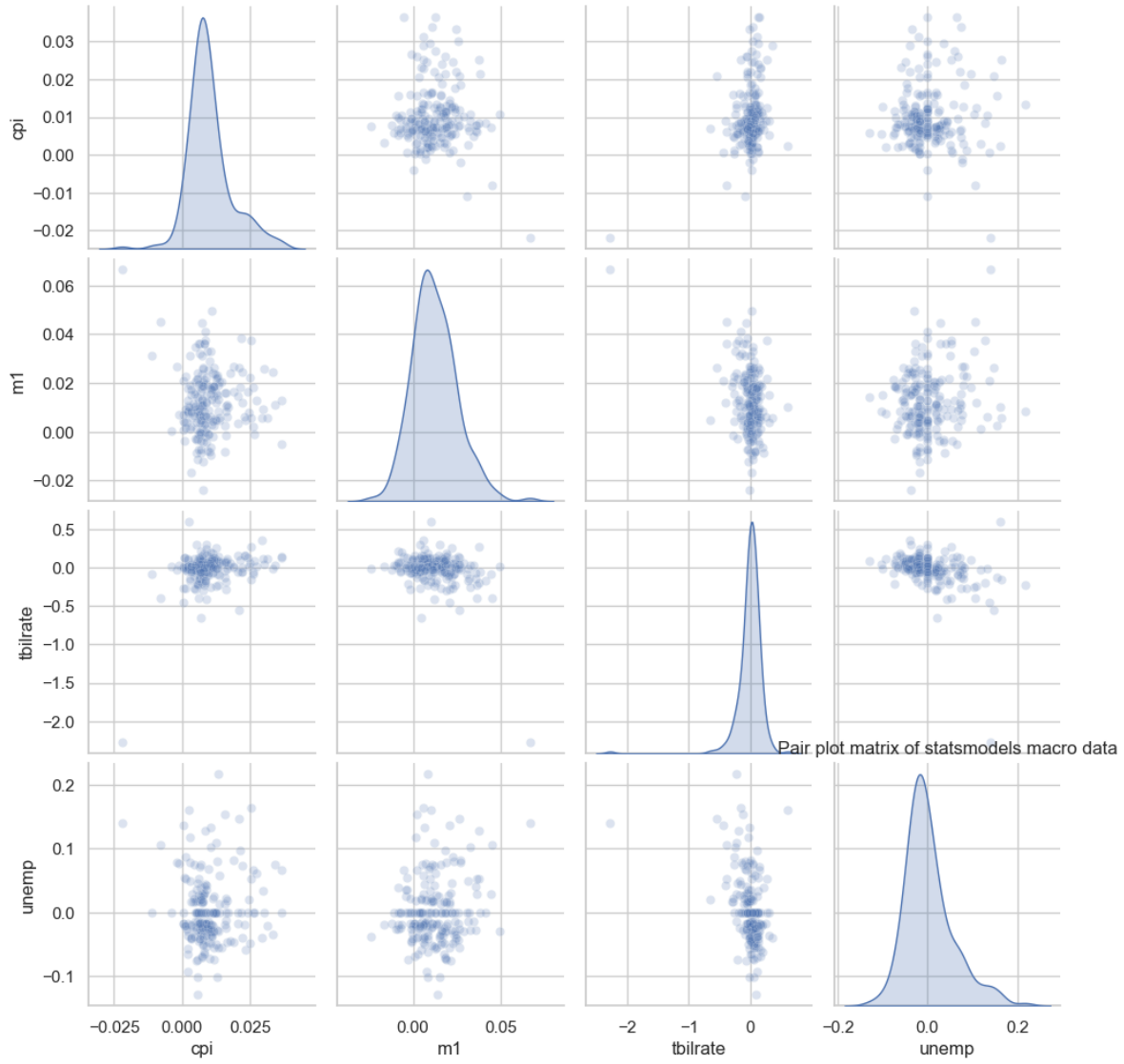
In [36]:
```
#点图或散点图可以用于检验两个一维数据序列之间的关系。
#比如从statsmodels项目中载入了macrodata数据集，并选择了一些变量，之后计算对数差:

macro = pd.read_csv('C:/Users/miran/lpthw/macrodata.csv')
macro
# year  quarter realgdp realcons    realinv realgovt    realdpi cpi m1  tbilra
# 0 1959    1   2710.349    1707.4  286.898 470.045 1886.9  28.980  139.7   2.
# 1 1959    2   2778.801    1733.7  310.859 481.301 1919.7  29.150  141.7   3.
# 2 1959    3   2775.488    1751.8  289.226 491.260 1916.4  29.350  140.5   3.
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
data
# cpi   m1  tbilrate    unemp
# 0 28.980  139.7   2.82    5.8
# 1 29.150  141.7   3.08    5.1
# 2 29.350  140.5   3.82    5.3

#对data进行对数变换，并计算对数变换后的差分，然后删除结果中的缺失值，最后将得到的新数
#.diff()方法计算对数变换后的差分，即计算每个元素与前一个元素的差值。
trans_data = np.log(data).diff().dropna()
trans_data
# cpi   m1  tbilrate    unemp
# 1 0.005849    0.014215    0.088193    -0.128617
# 2 0.006838    -0.008505   0.215321    0.038466
trans_data[-5:]                                             #选择变量trans_data
# 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，可以绘制散点图，并绘出一个条线性回归线。
sns.regplot(x = 'm1', y = 'unemp', data = trans_data)                   #注意
plt.title('Regression or Scatter Plot- Changes in log %s versus log %s' % ('m1
#A seaborn regression/scatter plot
```

Out[36]:
```
Text(0.5, 1.0, 'Regression or Scatter Plot- Changes in log m1 versus log unem
p')
```


Regression or Scatter Plot- Changes in log m1 versus log unemp
unemp
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.02
0.00
0.02
0.04
0.06
m1

In [37]:
```
#在探索性数据分析中，能够查看一组变量中的所有散点图有帮助的，被称为成对图或散点图矩阵。
#seaborn有个方便的pairplot函数，用于绘制多变量之间关系的函数。它会创建一个网格图，显
#它会创建一个网格图，显示数据集中多个变量之间的散点图和直方图。

# diag_kind='kde'：指定对角线上绘制的图形类型为核密度估计（Kernel Density Estimate
# plot_kws={'alpha': 0.2}：设置绘图的其他关键字参数，这里使用 alpha 参数将散点图的

sns.pairplot(trans_data, diag_kind = 'kde', plot_kws = {'alpha': 0.2})
plt.title('Pair plot matrix of statsmodels macro data')
#plot_ksw参数，使我们能够将配置选项传递给非对角元素上的各个绘图调用。
```

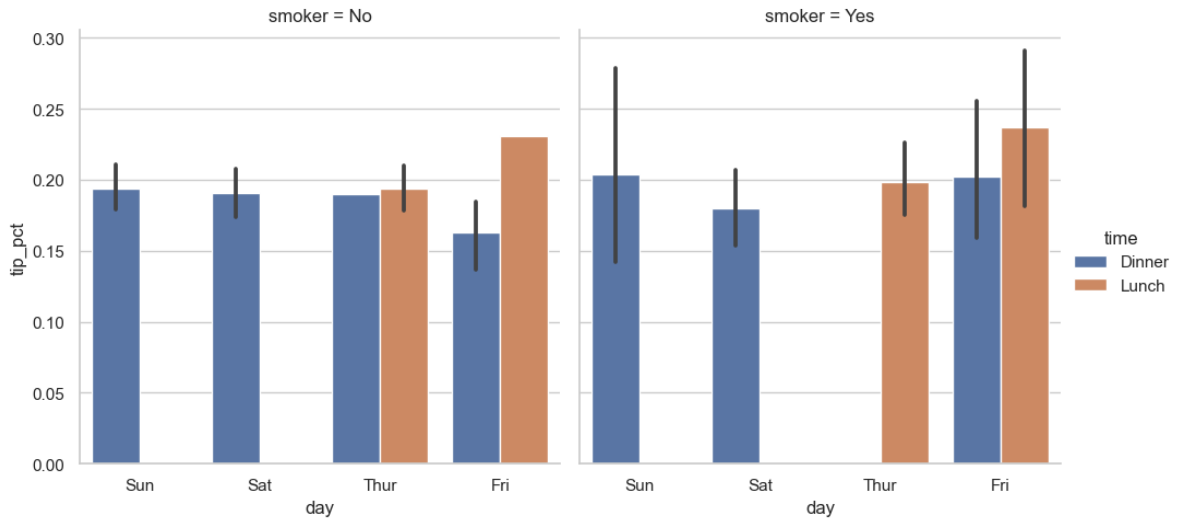
Out[37]: Text(0.5, 1.0, 'Pair plot matrix of statsmodels macro data')



9.2.5 分面网格和分类数据 Facet Grids and Categorical Data

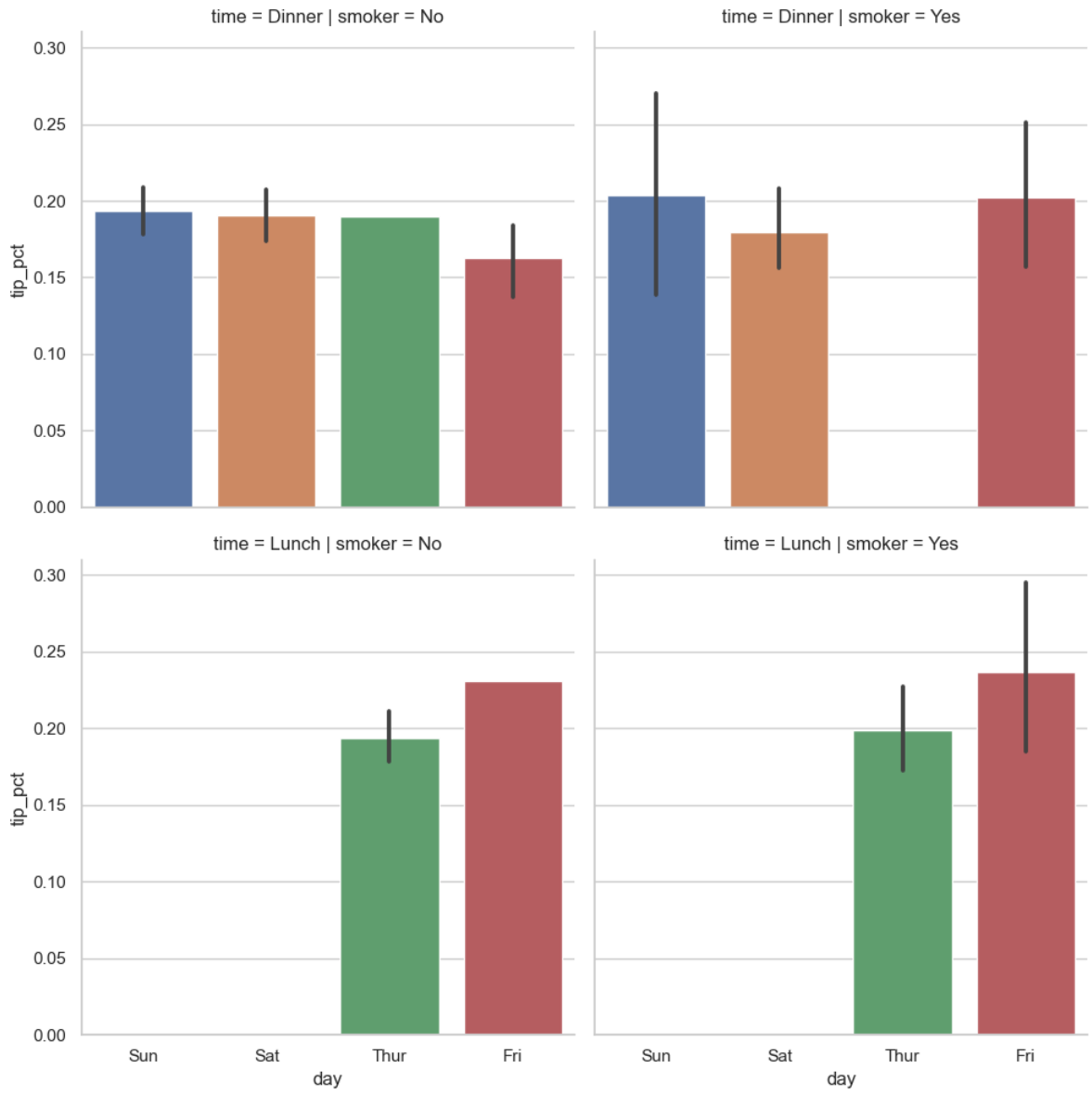
In [38]:
```
#如果数据集有额外的分组维度，可以使用分面网格利用多种分组变量，对数据进行可视化。hue
#seaborn拥有一个有效的内建函数factorplot，可以简化多种分面绘图:
#在新版的 Seaborn 库中，factorplot() 已被弃用，并替换为更灵活的 catplot() 函数。书
#柱子上的黑线，代表95%置信区间。

# x='day': 指定在 x 轴上显示的变量为 'day'，表示天数。
# y='tip_pct': 指定在 y 轴上显示的变量为 'tip_pct'，表示小费百分比。
# hue='time': 根据 'time' 变量对数据进行分类，使用不同颜色表示不同的时间段。
# col='smoker': 根据 'smoker' 变量对数据进行分列，生成不同的列来展示吸烟者和非吸烟者
# kind='bar': 指定绘图类型为条形图，以柱状图的形式展示数据。
# data=tips[tips.tip_pct < 1]: 指定使用的数据集为 tips，并使用筛选条件 tips.tip_p

sns.catplot(x = 'day', y = 'tip_pct', hue = 'time', col = 'smoker',
               kind = 'bar', data = tips[tips.tip_pct < 1])
#plt.title('Tipping percentage by day/time/smoker')
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x1bc0030a890>



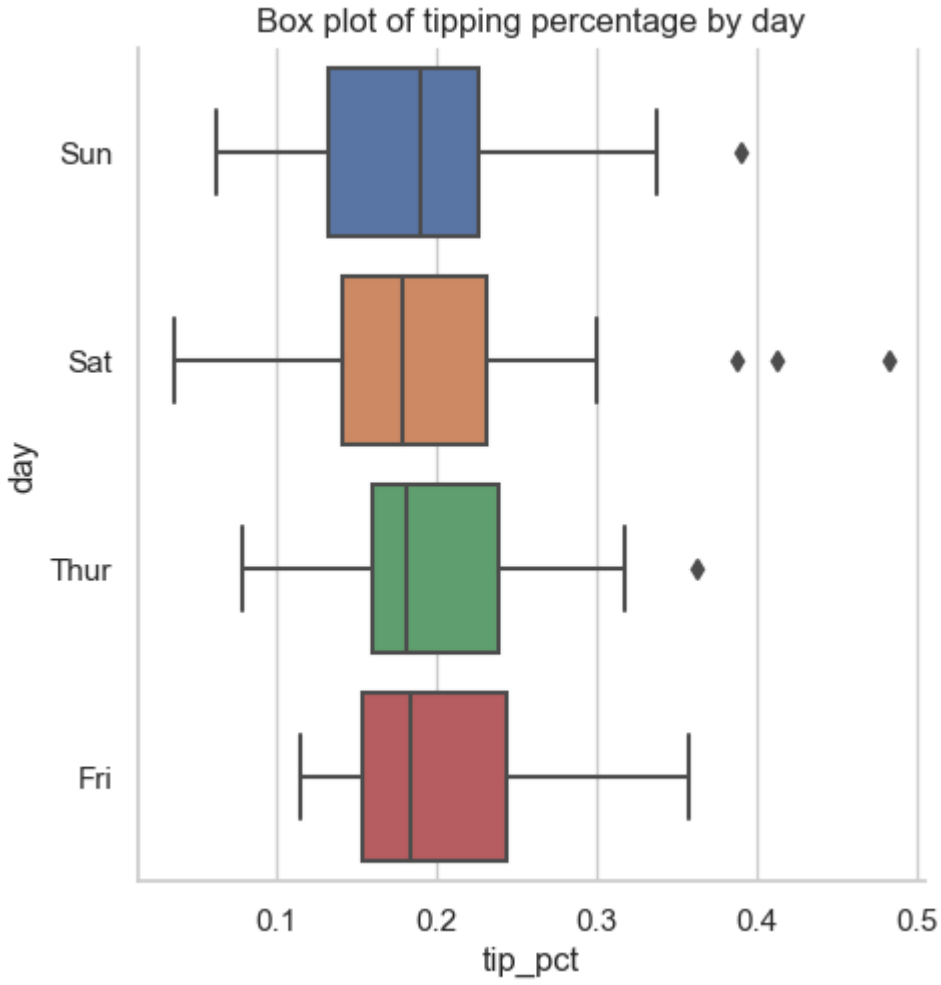
In [39]:
```
#除了根据time 在一个面内，将不同的柱分组为不同的颜色，还可以通过每个时间值添加一行来

# x='day'：指定在 x 轴上显示的变量为 'day'，表示天数。
# y='tip_pct'：指定在 y 轴上显示的变量为 'tip_pct'，表示小费百分比。
# row='time'：根据 'time' 变量对数据进行分行，生成不同的行来展示不同的时间段。
# col='smoker'：根据 'smoker' 变量对数据进行分列，生成不同的列来展示吸烟者和非吸烟
# kind='bar'：指定绘图类型为条形图，以柱状图的形式展示数据。
# data=tips[tips.tip_pct < 1]：指定使用的数据集为 tips，并使用筛选条件 tips.tip_p

sns.catplot(x = 'day', y = 'tip_pct', row = 'time', col = 'smoker',
            kind = 'bar', data = tips[tips.tip_pct < 1])
#plt.title('Tipping percentage by day split by time/smoker')
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x1bc03955f00>



In [40]:
```
#catplot也支持箱线图（中位值，四分位，异常值）可以是有效的可视化类型：
sns.catplot(x = 'tip_pct', y = 'day', kind = 'box',
            data = tips[tips.tip_pct < 0.5])
plt.title('Box plot of tipping percentage by day')
```

Out[40]:
```
Text(0.5, 1.0, 'Box plot of tipping percentage by day')
```



9.3 其他python可视化工具 Other Python Visualization Tools

In [41]:
```
#https://altair-viz.github.io/
#https://docs.bokeh.org/en/latest/
#https://plotly.com/python/
#https://github.com/plotly/plotly.py
```

9.4 本章小结Conclusion