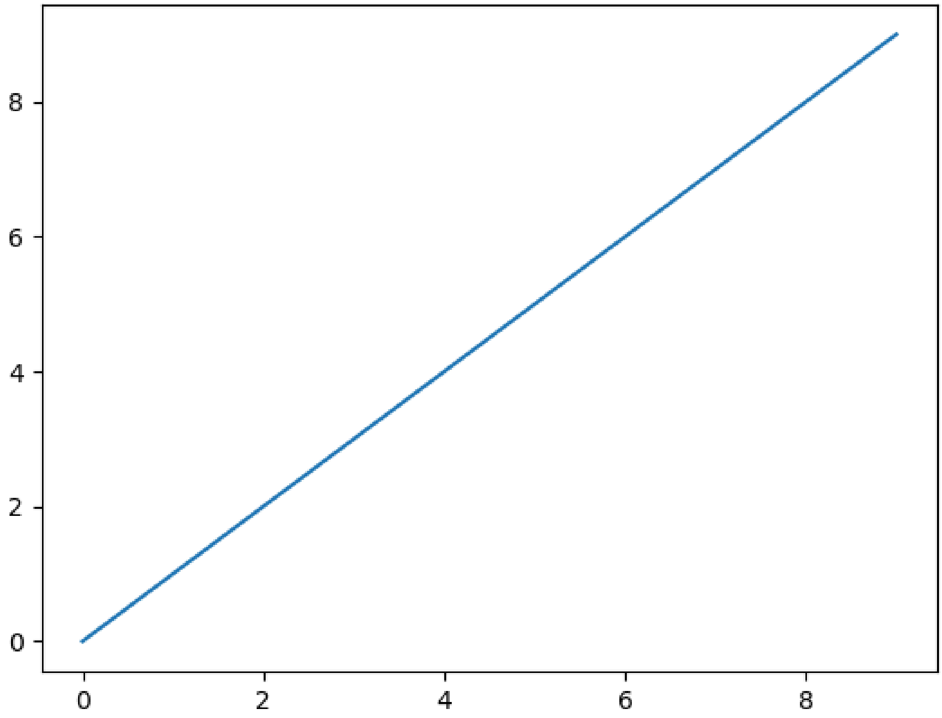
Plotting and Visualization

In [1]:
```
%matplotlib notebook
```

A Brief matplotlib API Primer

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
import numpy as np
data = np.arange(10)
data
plt.plot(data)
```



Out[3]:
```
[<matplotlib.lines.Line2D at 0x2214056a550>]
```

Figures and Subplots

In [4]:
```
fig = plt.figure()
```

In [5]:

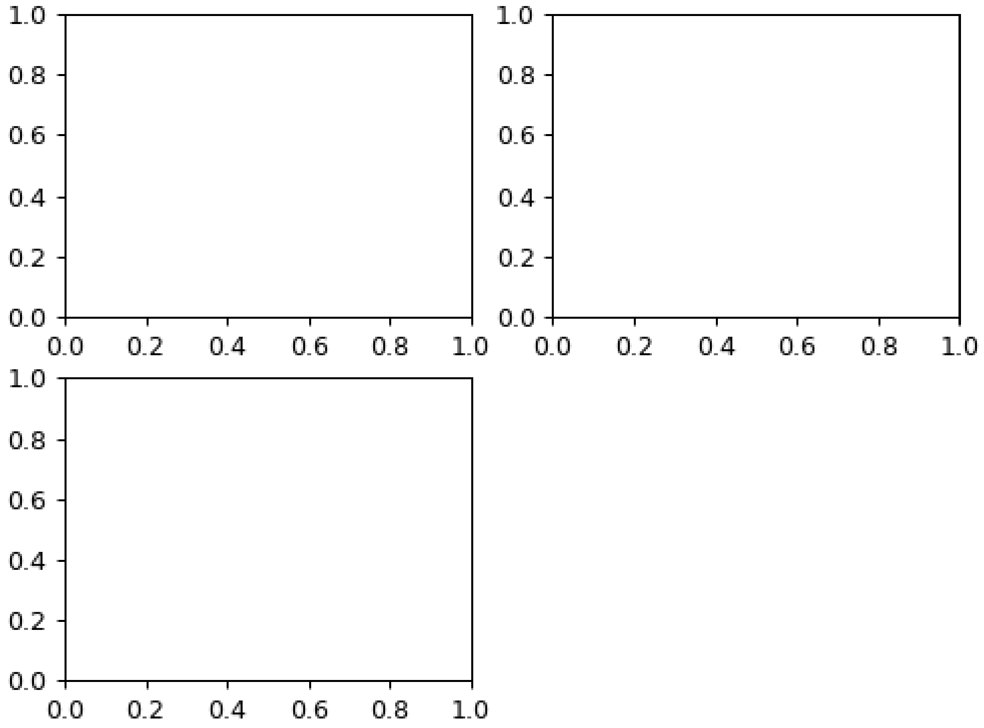
```
ax1 = fig.add_subplot(2, 2, 1)
```

In [6]:

```
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```

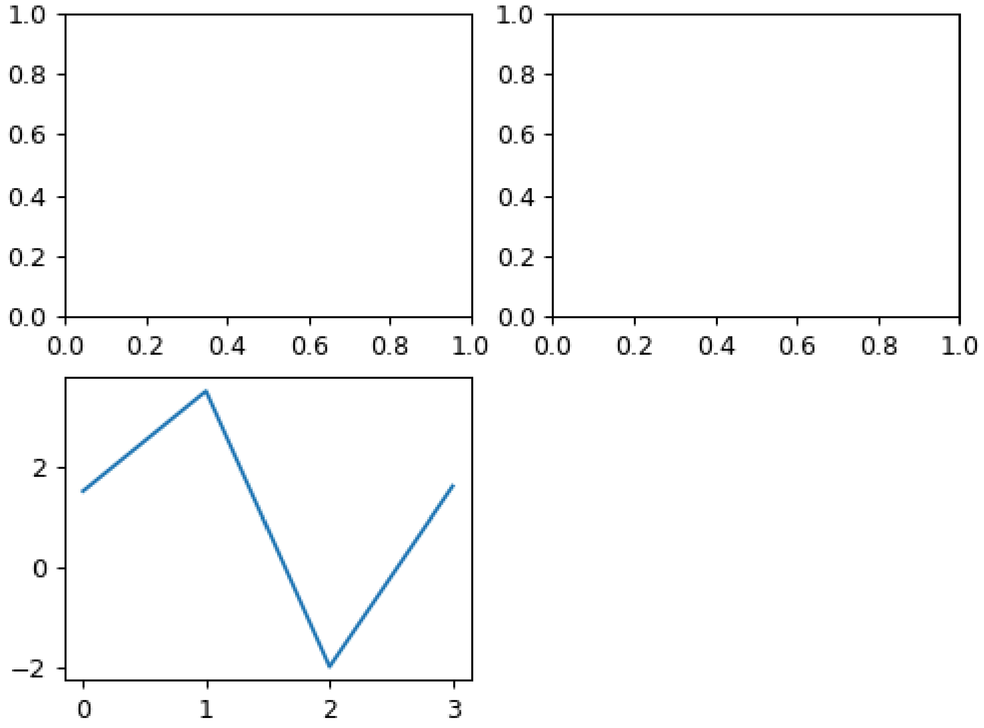
In [7]:

```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```



In [10]:

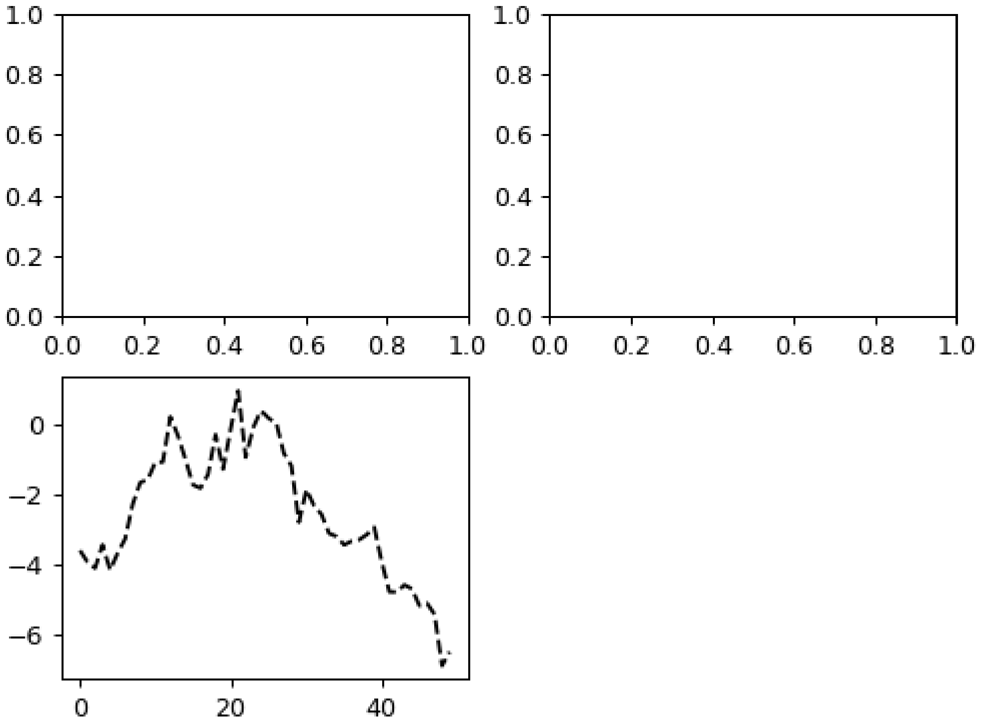
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
plt.plot([1.5, 3.5,-2, 1.6])
```



Out[10]:

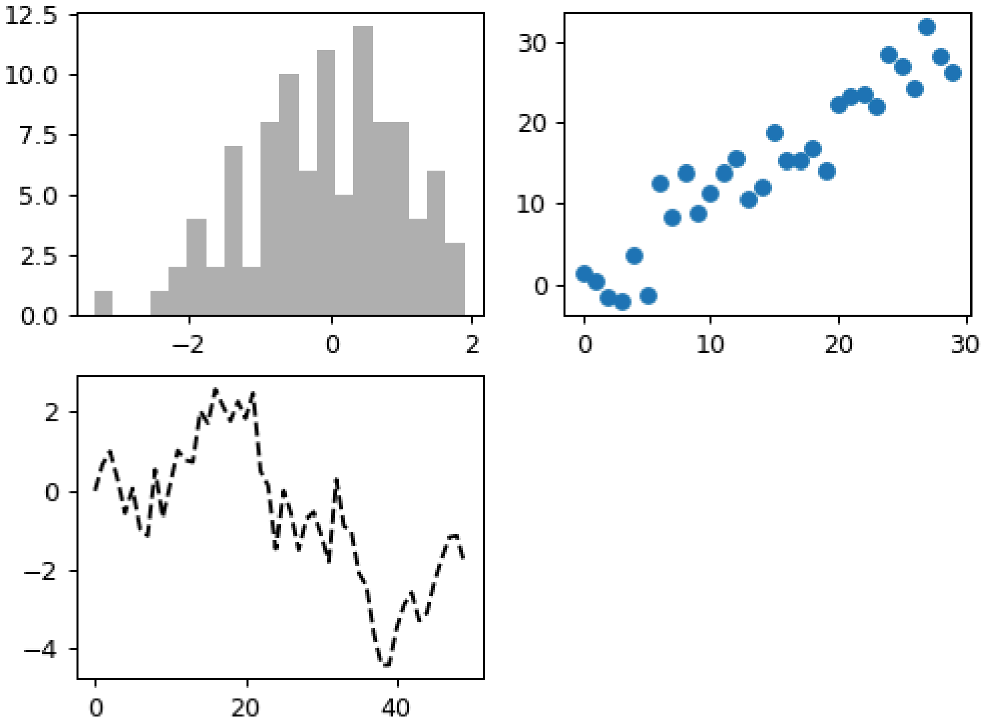
```
[<matplotlib.lines.Line2D at 0x22140b20d00>]
```

In [11]:
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
plt.plot(np.random.randn(50).cumsum(), 'k--')
```



Out[11]:
```
[<matplotlib.lines.Line2D at 0x22141bfb490>]
```

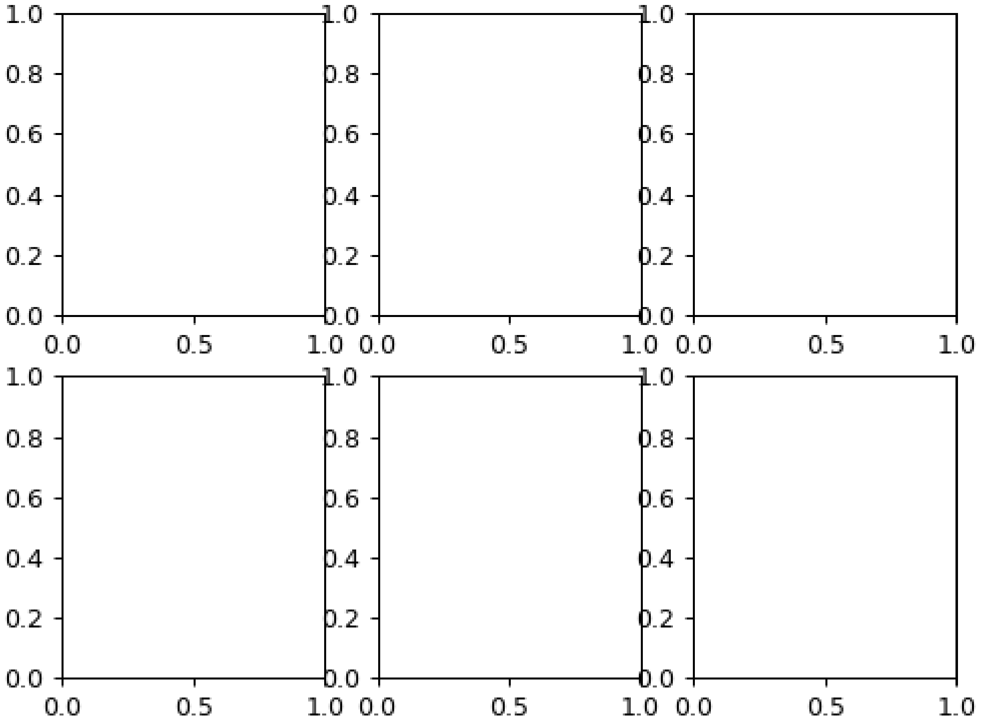
In [13]:
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
plt.plot(np.random.randn(50).cumsum(), 'k--')
_ = ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
```



Out[13]:
```
<matplotlib.collections.PathCollection at 0x22141c9d1f0>
```

In [14]:
```
plt.close('all')
```

In [15]:
```
fig, axes = plt.subplots(2, 3)
axes
```

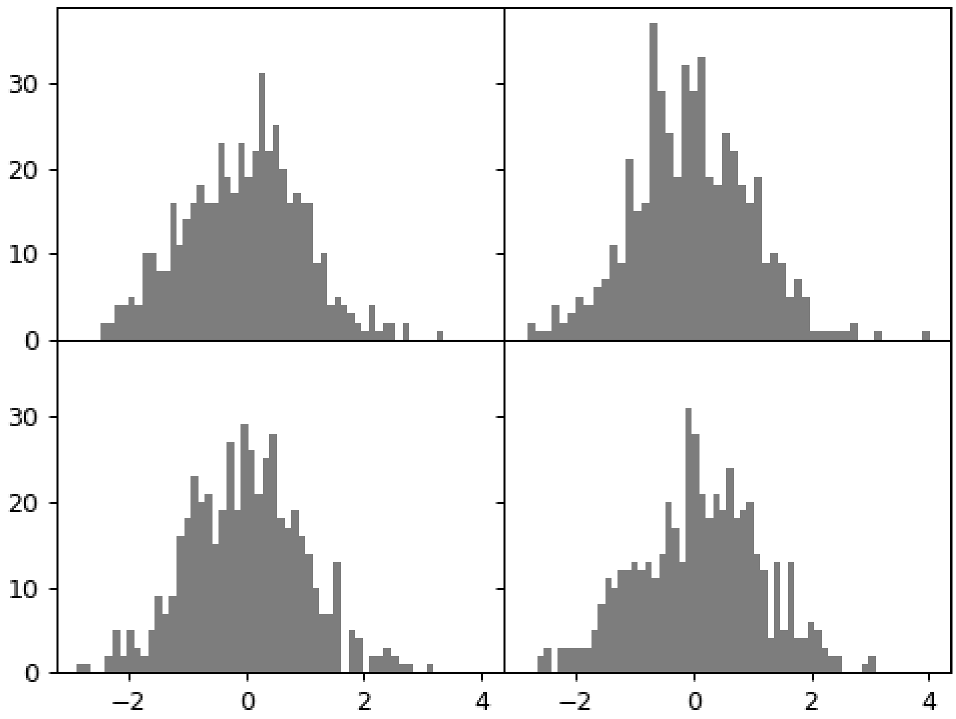


Out[15]:
```
array([[<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```

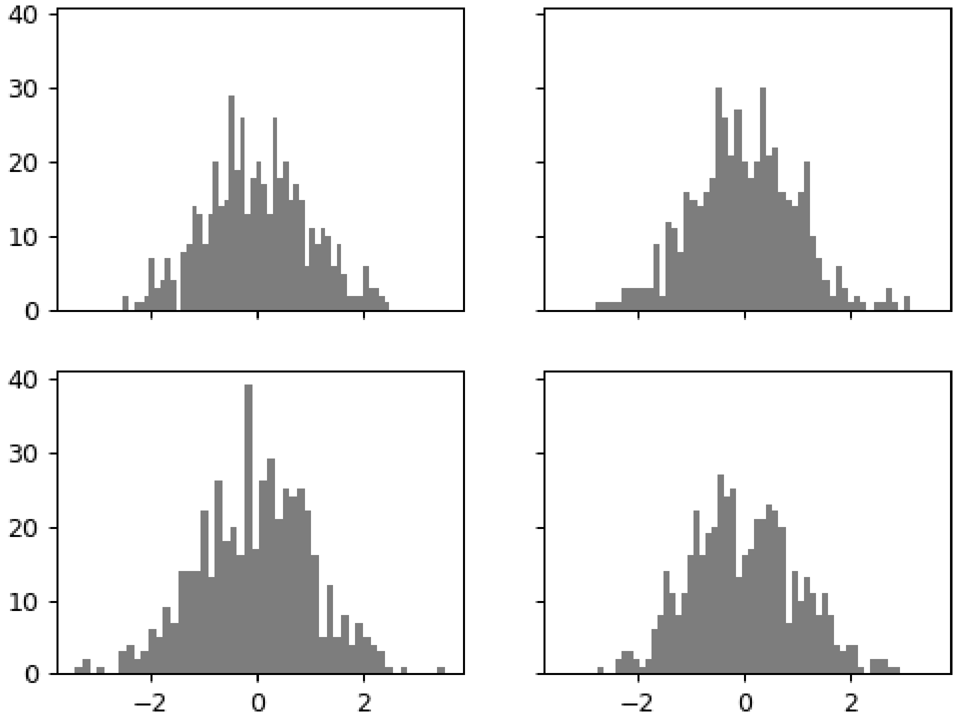
Adjusting the spacing around subplots

In [17]:
```
plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
                wspace=None, hspace=None)
```

In [18]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
```

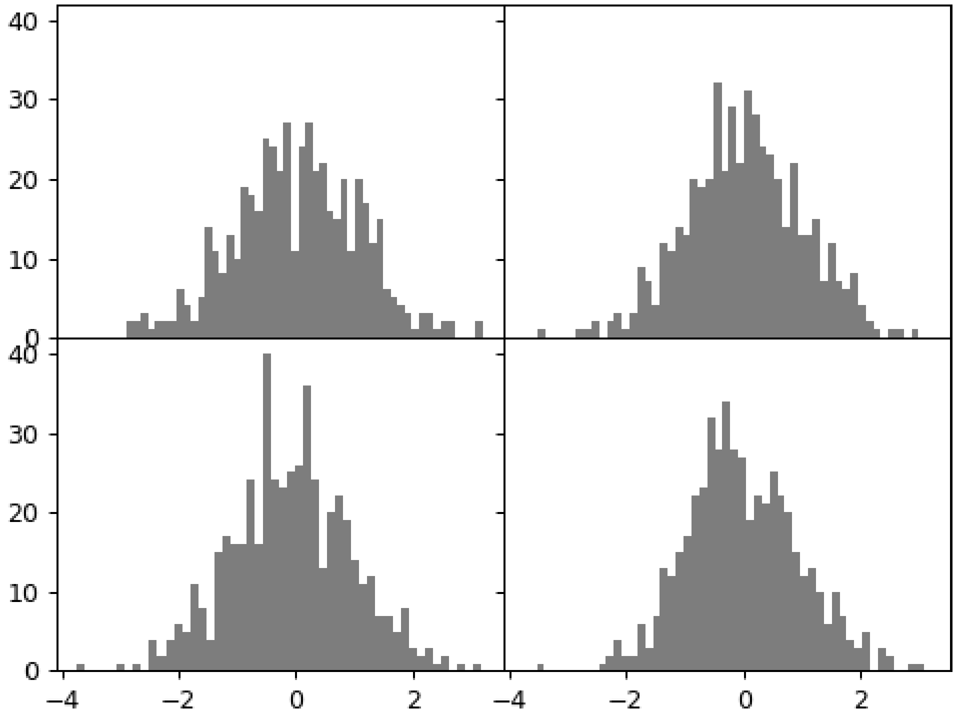


In [19]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust()
```



In [20]:

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



Colors, Markers, and Line Styles

ax.plot(x, y, 'g--')

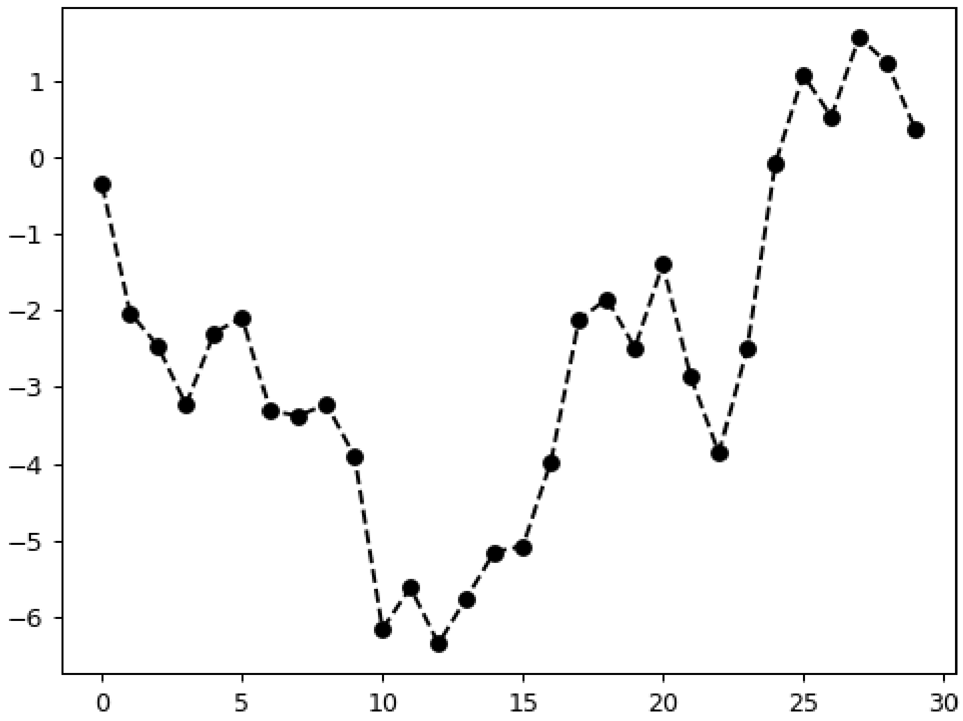
ax.plot(x, y, linestyle='--', color='g')

In [22]:

```
plt.figure()
```

In [24]:

```
from numpy.random import randn
plt.figure()
plt.plot(randn(30).cumsum(), 'ko--')
```

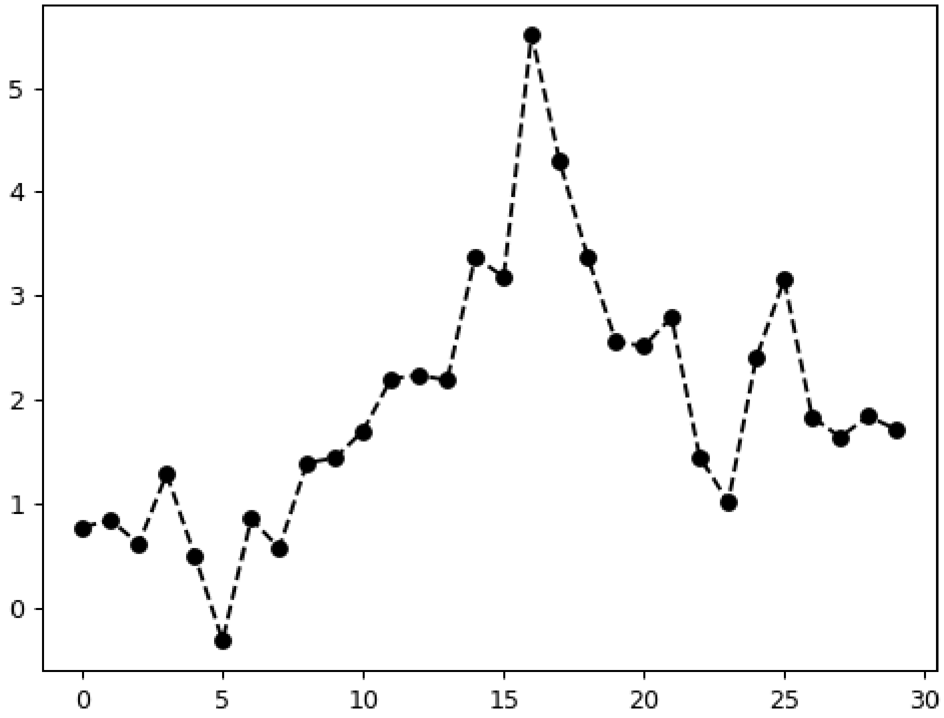


Out[24]:

```
[<matplotlib.lines.Line2D at 0x221438a3490>]
```

In [26]:

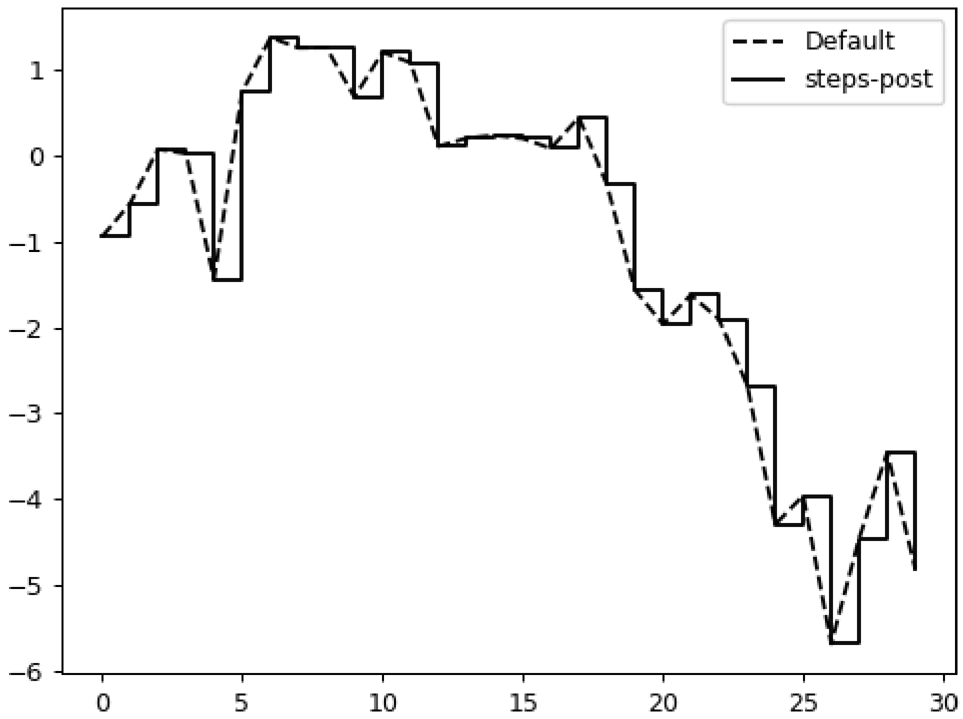
```
plt.figure()
plt.plot(randn(30).cumsum(), color='k', linestyle='dashed', marker='o')
```



Out[26]:
```
[<matplotlib.lines.Line2D at 0x22143912700>]
```

In [27]:
```
plt.close('all')
```

In [28]:
```
data = np.random.randn(30).cumsum()
plt.plot(data, 'k--', label='Default')
plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')
```



Out[28]:

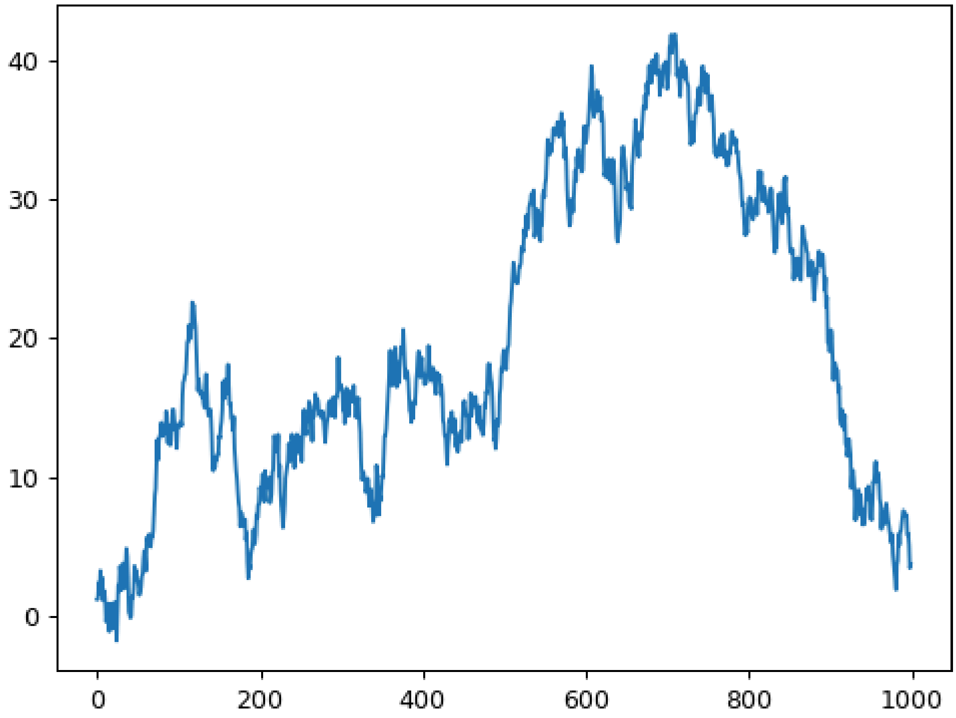
```
<matplotlib.legend.Legend at 0x22142544370>
```

Ticks, Labels, and Legends

Setting the title, axis labels, ticks, and ticklabels

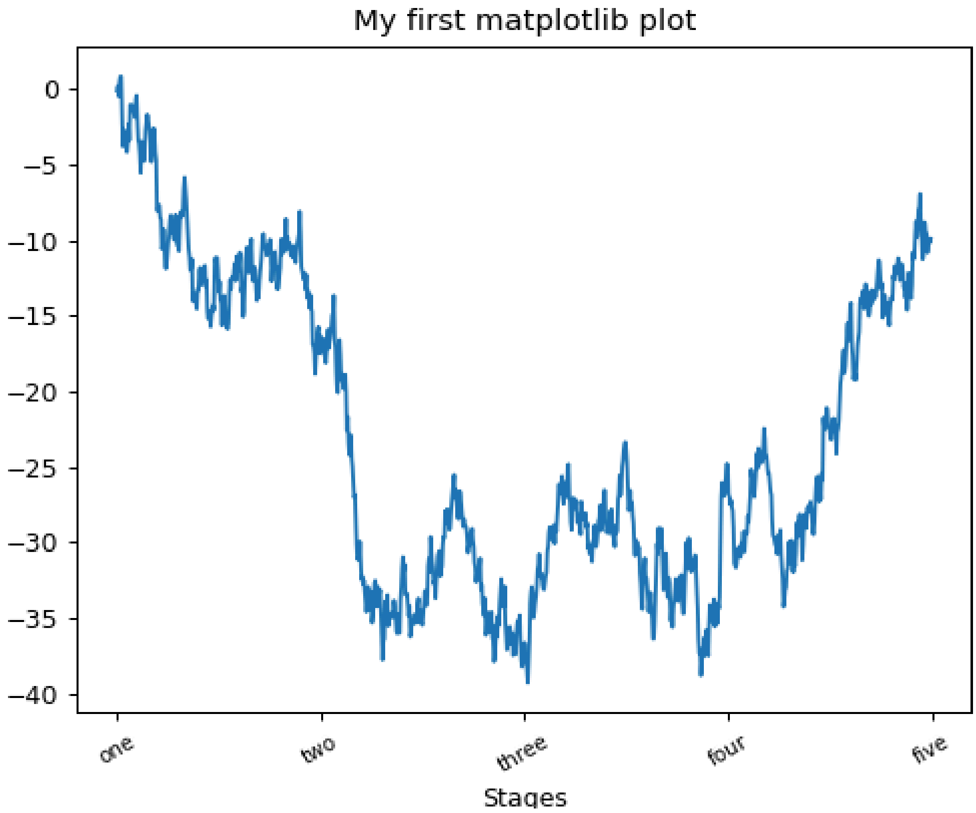
In [29]:

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



Out[29]:
```
[<matplotlib.lines.Line2D at 0x2214252eaf0>]
```

In [34]:
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(np.random.randn(1000).cumsum())

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')
```



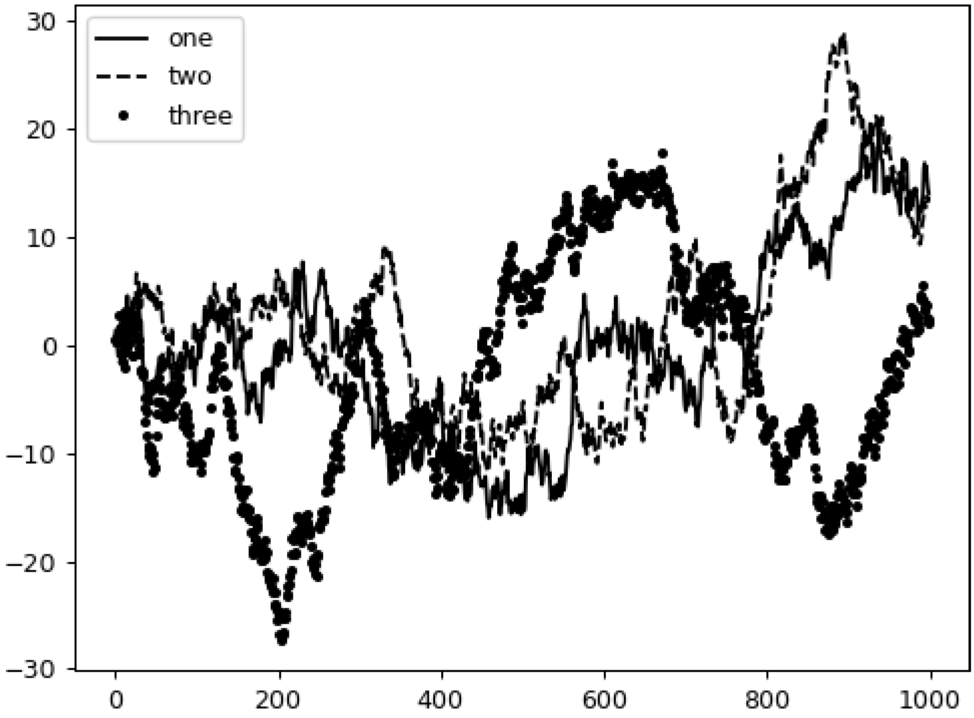
Out[34]:
```
Text(0.5, 0, 'Stages')
```

In [35]:
```
props = {
    'title': 'My first matplotlib plot',
    'xlabel': 'Stages'
}
ax.set(**props)
```

Out[35]:
```
[Text(0.5, 1.0, 'My first matplotlib plot'),
 Text(0.5, 11.297245067556123, 'Stages')]
```

Adding legends

In [36]:
```
from numpy.random import randn
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'k', label='one')
ax.plot(randn(1000).cumsum(), 'k--', label='two')
ax.plot(randn(1000).cumsum(), 'k.', label='three')
ax.legend(loc='best')
```

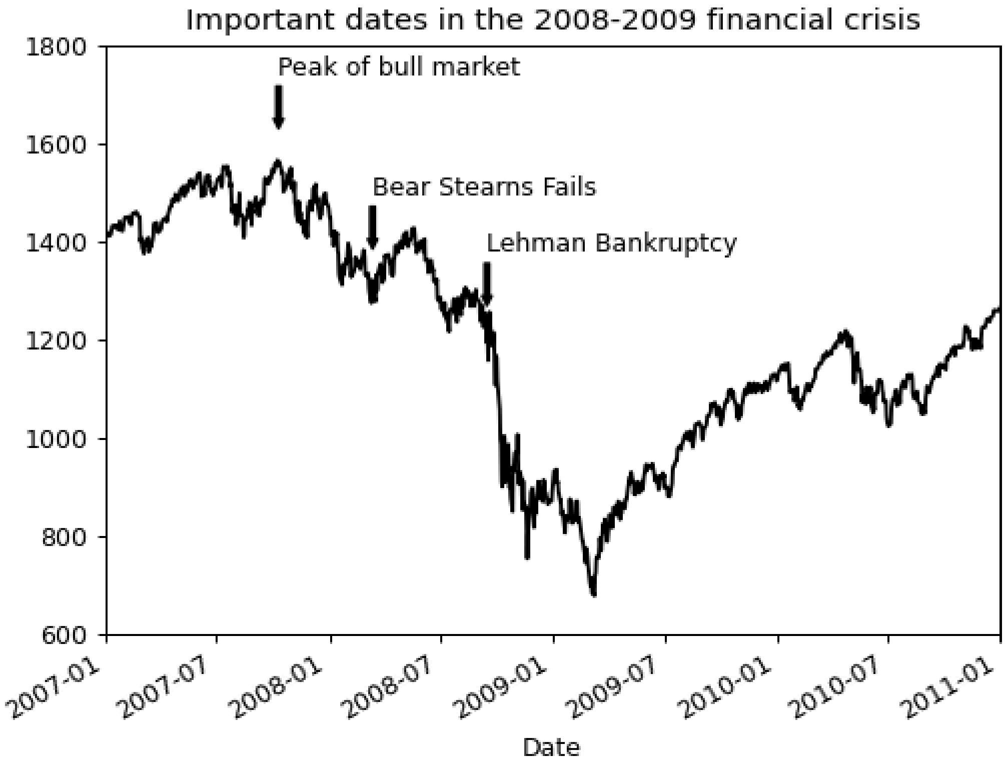


Out[36]:
```
<matplotlib.legend.Legend at 0x22142281a60>
```

Annotations and Drawing on a Subplot

In [ ]:
```
ax.text(x, y, 'Hello world!',
        family='monospace', fontsize=10)
```

In [39]:
```
from datetime import datetime
import pandas as pd

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

data = pd.read_csv('examples/spx.csv', index_col=0, parse_dates=True)
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, width=2,
                                headlength=4),
                horizontalalignment='left', verticalalignment='top')

# Zoom in on 2007-2010
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')
```

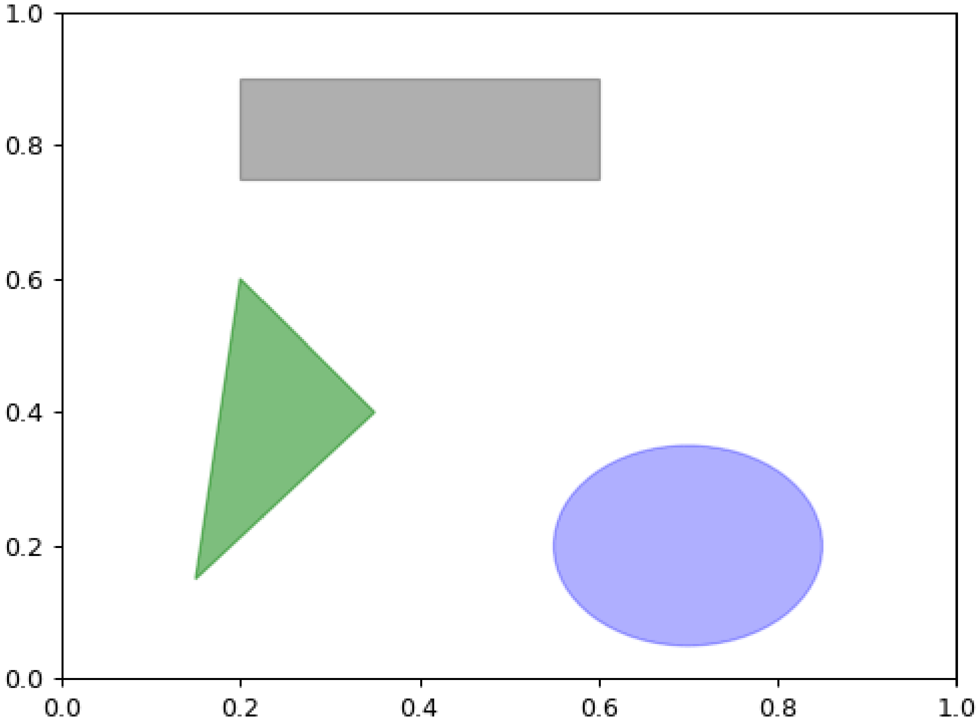


Out[39]:
```
Text(0.5, 1.0, 'Important dates in the 2008-2009 financial crisis')
```

In [41]:
```
ax.set_title('Important dates in the 2008–2009 financial crisis')
```

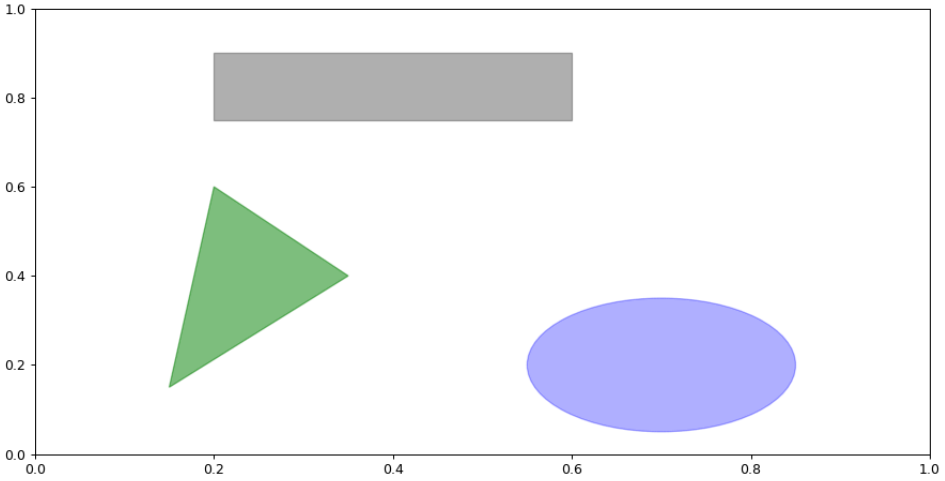
Out[41]:
```
Text(0.5, 1.0, 'Important dates in the 2008–2009 financial crisis')
```

In [40]:
```
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)
```



Out[40]:
```
<matplotlib.patches.Polygon at 0x22140896a90>
```

In [42]:
```
fig = plt.figure(figsize=(12, 6)); 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)
```



Out[42]:
```
<matplotlib.patches.Polygon at 0x221454b0e80>
```

Saving Plots to File

In [43]:

```
plt.savefig('figpath.svg')
```

In [44]:

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

In [45]:

```
from io import BytesIO
buffer = BytesIO()
plt.savefig(buffer)
plot_data = buffer.getvalue()
```

matplotlib Configuration

In [46]:

```
plt.rc('figure', figsize=(10, 10))
```

In [47]:

```
font_options = {'family' : 'monospace',
                'weight' : 'bold',
                'size'   : 'small'}
plt.rc('font', **font_options)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1612\1999892431.py in <module>
      2                 'weight' : 'bold',
      3                 'size'   : 'small'}
----> 4 plt.rc('font', **font_options)

C:\PythonDSA\anaconda3\lib\site-packages\matplotlib\pyplot.py in rc(group, **kwarg
s)
    574 @_copy_docstring_and_deprecators(matplotlib.rc)
    575 def rc(group, **kwargs):
--> 576     matplotlib.rc(group, **kwargs)
    577
    578

C:\PythonDSA\anaconda3\lib\site-packages\matplotlib\__init__.py in rc(group, **kwa
rgs)
    973             key = '%s.%s' % (g, name)
    974             try:
--> 975                 rcParams[key] = v
    976             except KeyError as err:
    977                 raise KeyError(('Unrecognized key "%s" for group "%s" and
'

C:\PythonDSA\anaconda3\lib\site-packages\matplotlib\__init__.py in __setitem__(sel
f, key, val)
    644                 cval = self.validate[key](val)
    645             except ValueError as ve:
--> 646                 raise ValueError(f"Key {key}: {ve}") from None
    647             dict.__setitem__(self, key, cval)
    648         except KeyError as err:

ValueError: Key font.size: Could not convert 'small' to float
```

Plotting with pandas and seaborn

Line Plots

In [1]:

```
plt.close('all')
```

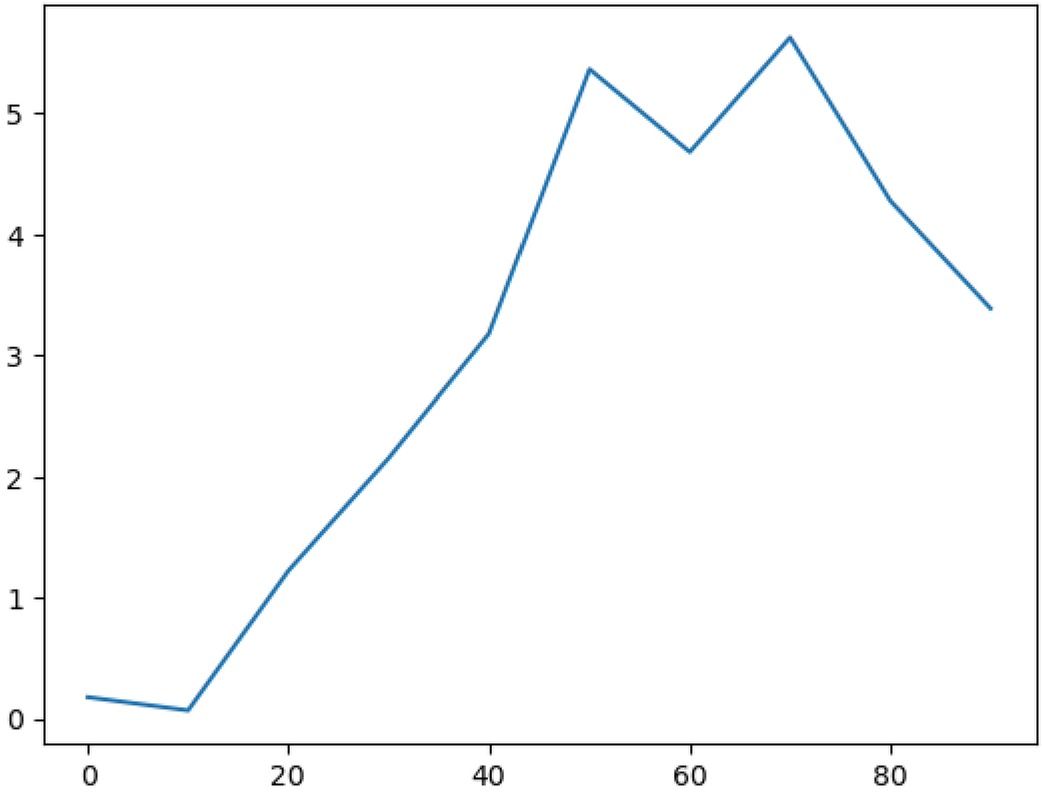
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8880\765339104.py in <module>
----> 1 plt.close('all')

NameError: name 'plt' is not defined
```

In [4]:

```
import pandas as pd
import numpy as np
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
s.plot()
```

Out[4]:

```
<AxesSubplot:>
```

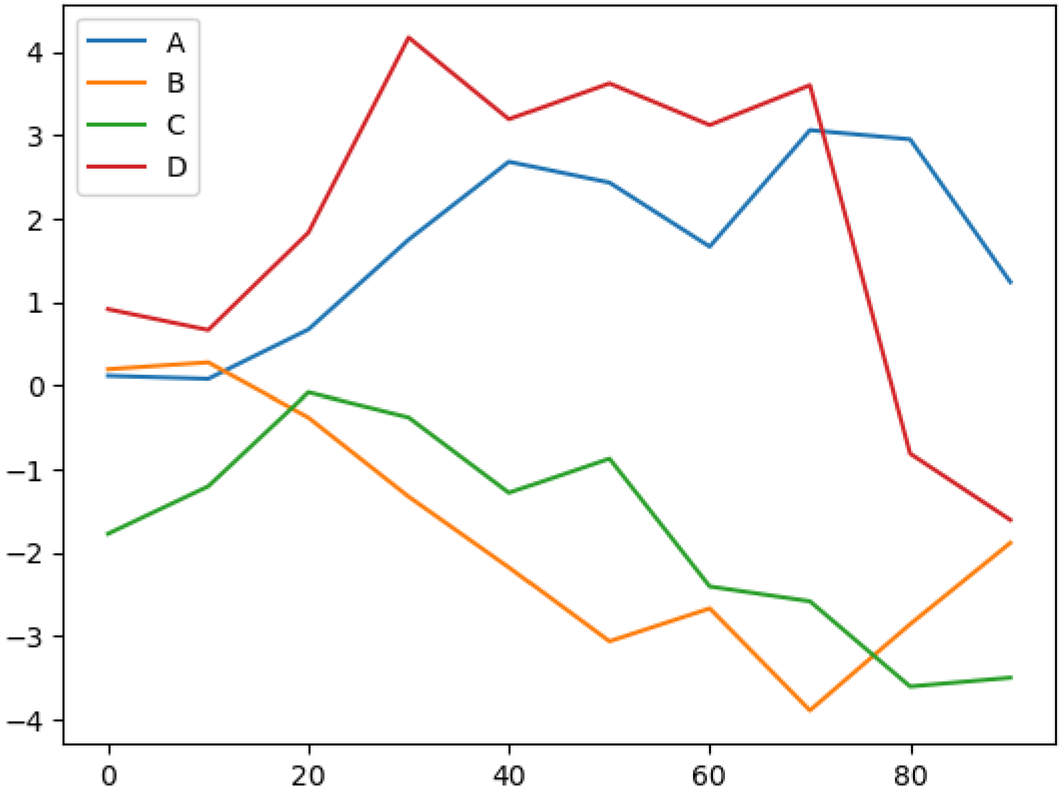


In [5]:

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

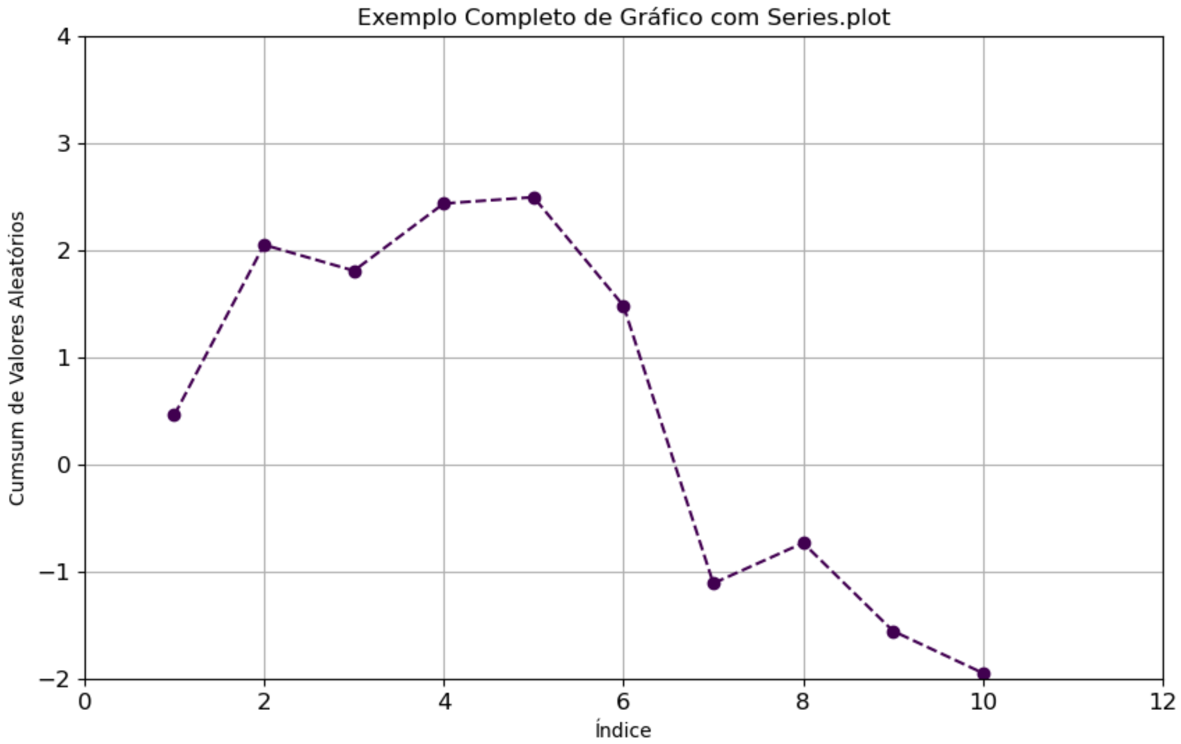
Out[5]:

```
<AxesSubplot:>
```



In [8]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Criando uma Series de exemplo
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(1, 11))

# Configurando o gráfico com todos os argumentos
s.plot(kind='line',              # Tipo de gráfico (linha)
       ax=None,                  # Usa um novo objeto Axes
       figsize=(10, 6),          # Tamanho da figura (10x6 polegadas)
       use_index=True,           # Usa o índice como eixo x
       title='Exemplo Completo de Gráfico com Series.plot',  # Título do gráfico
       grid=True,                # Mostra a grade
       legend=False,             # Não exibe a legenda
       style='--o',              # Linha tracejada com marcadores circulares
       logx=False,               # Não usa escala logarítmica no eixo x
       logy=False,               # Não usa escala logarítmica no eixo y
       loglog=False,             # Não usa escala logarítmica em ambos os eixos
       xlim=(0, 12),             # Limites do eixo x
       ylim=(-2, 4),             # Limites do eixo y
       rot=0,                    # Não rotaciona os rótulos do eixo x
       fontsize=12,              # Tamanho da fonte dos rótulos dos eixos
       colormap='viridis',       # Mapa de cores (não aplicável a gráficos de linha
       table=False,              # Não desenha uma tabela
       )

# Exibindo o gráfico
plt.xlabel('Índice')
plt.ylabel('Cumsum de Valores Aleatórios')
plt.show()
```



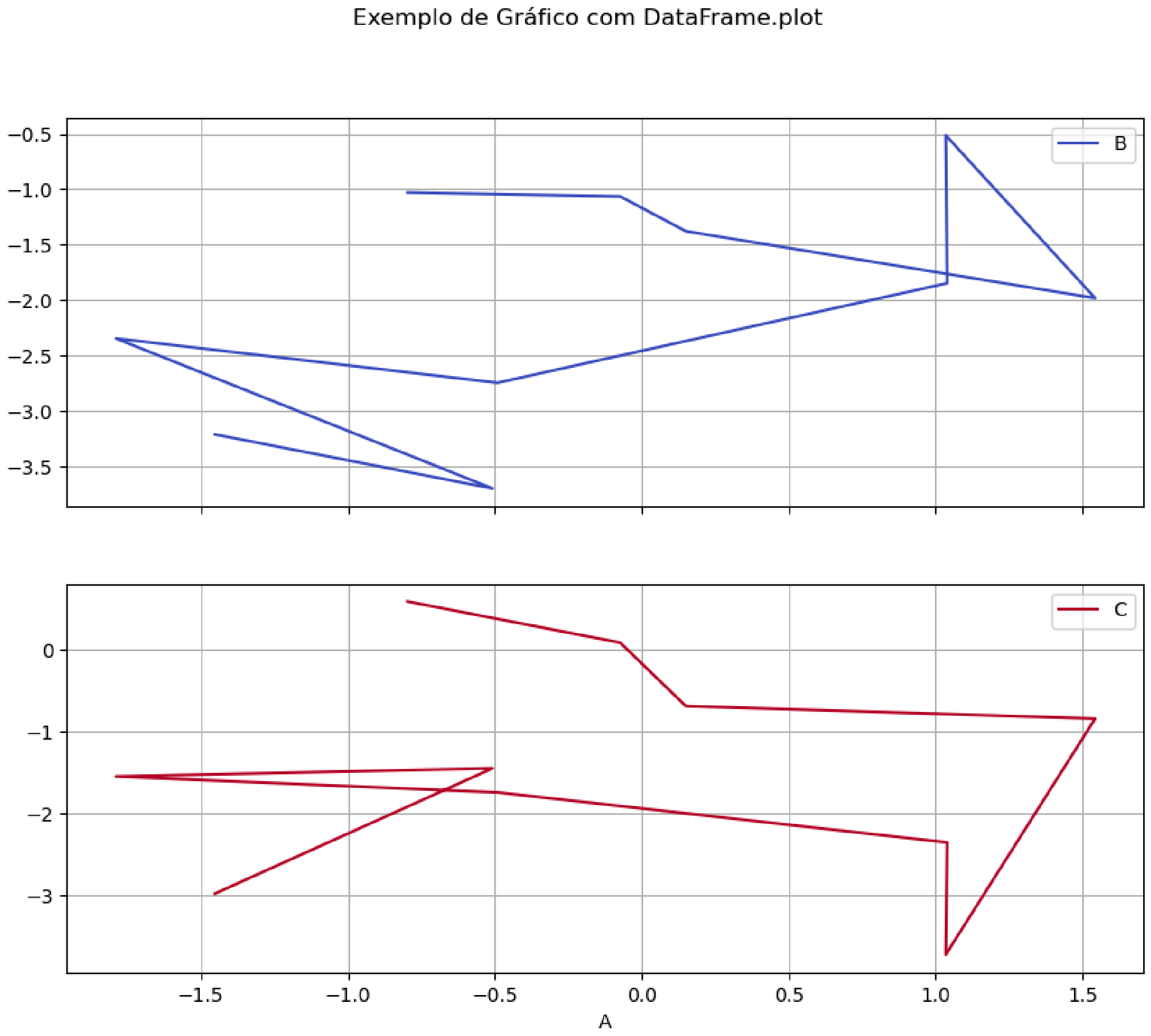
In [ ]:

```
# Explicação dos Argumentos Utilizados:
kind='line': Cria um gráfico de linha (padrão).
ax=None: Cria um novo objeto Axes para o gráfico.
figsize=(10, 6): Define o tamanho da figura em 10 polegadas de largura e 6 de altur
use_index=True: Usa o índice da série (valores de 1 a 10) no eixo x.
title='Exemplo Completo de Gráfico com Series.plot': Define o título do gráfico.
grid=True: Exibe uma grade no gráfico.
legend=False: Não exibe legenda, já que estamos plotando uma única série.
style='--o': Define o estilo da linha como tracejada com marcadores circulares.
logx=False, logy=False, loglog=False: Não usa escalas logarítmicas nos eixos.
xlim=(0, 12): Define os limites do eixo x entre 0 e 12.
ylim=(-2, 4): Define os limites do eixo y entre -2 e 4.
rot=0: Mantém os rótulos do eixo x não rotacionados.
fontsize=12: Define o tamanho da fonte dos rótulos dos eixos como 12.
colormap='viridis': Não aplicável ao gráfico de linha, mas mencionado para referênc
table=False: Não desenha uma tabela abaixo do gráfico.
```

In [9]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Criando um DataFrame de exemplo
df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                  columns=['A', 'B', 'C', 'D'],
                  index=np.arange(0, 100, 10))

# Plotando o DataFrame com argumentos específicos
df.plot(x='A', y=['B', 'C'],          # Especifica colunas para eixo x e y
        kind='line',                  # Tipo de gráfico
        subplots=True,                # Cria subgráficos
        sharex=True,                  # Subgráficos compartilham o eixo x
        layout=(2, 1),                # Layout de subgráficos (2 linhas, 1 coluna)
        figsize=(10, 8),              # Tamanho da figura
        title='Exemplo de Gráfico com DataFrame.plot',  # Título do gráfico
        grid=True,                    # Exibe a grade
        legend=True,                  # Exibe a legenda
        colormap='coolwarm',          # Mapa de cores
        secondary_y='D',              # Coluna 'D' em eixo y secundário
        mark_right=True)              # Marca o eixo y secundário na legenda
```

```
plt.show()
```



In [ ]:

```
Explicação do Exemplo:
x='A', y=['B', 'C']: Especifica a coluna 'A' para o eixo x e as colunas 'B' e 'C' p
subplots=True: Cria subgráficos separados para 'B' e 'C'.
sharex=True: Os subgráficos compartilham o eixo x.
layout=(2, 1): Organiza os subgráficos em 2 linhas e 1 coluna.
secondary_y='D': Plota a coluna 'D' em um eixo y secundário.
mark_right=True: Adiciona uma marca no eixo y secundário.
```

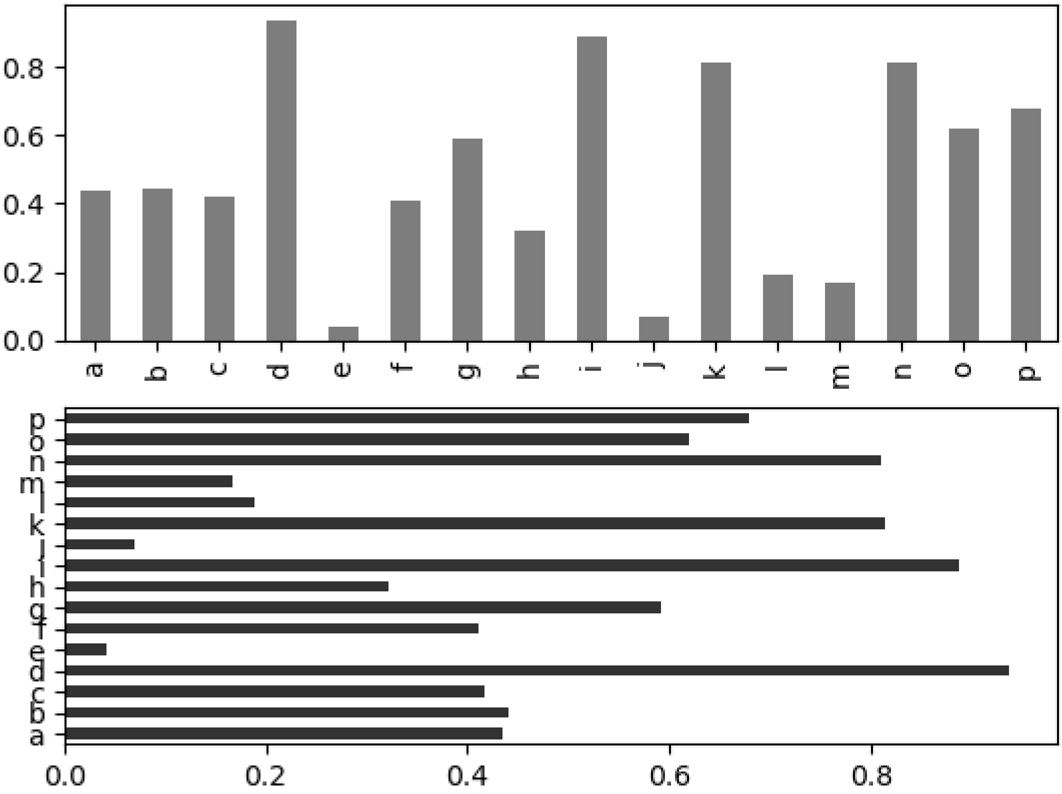
Bar Plots

In [11]:

```
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.5)
data.plot.barh(ax=axes[1], color='k', alpha=0.8)
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
np.random.seed(12348)
```

In [13]:

```
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
```

Out[13]:

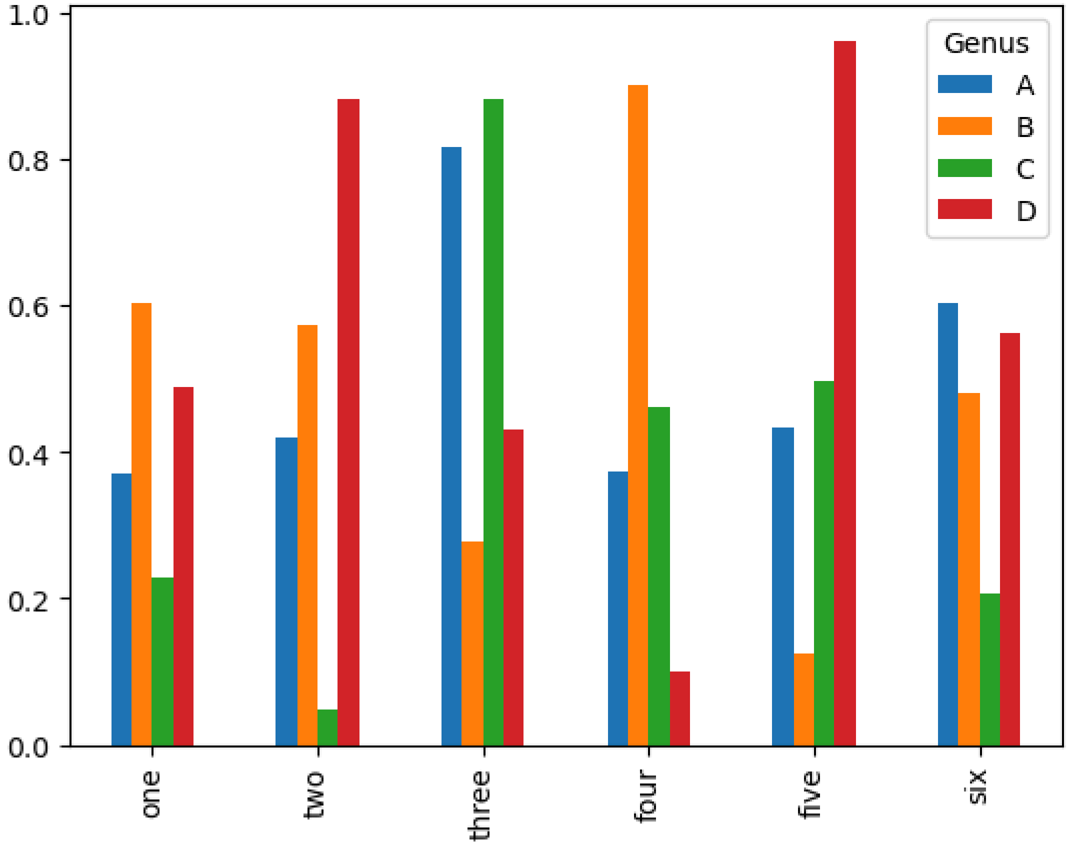
Genus	A	B	C	D
one	0.370670	0.602792	0.229159	0.486744
two	0.420082	0.571653	0.049024	0.880592
three	0.814568	0.277160	0.880316	0.431326
four	0.374020	0.899420	0.460304	0.100843
five	0.433270	0.125107	0.494675	0.961825
six	0.601648	0.478576	0.205690	0.560547

In [14]:

```
df.plot.bar()
```

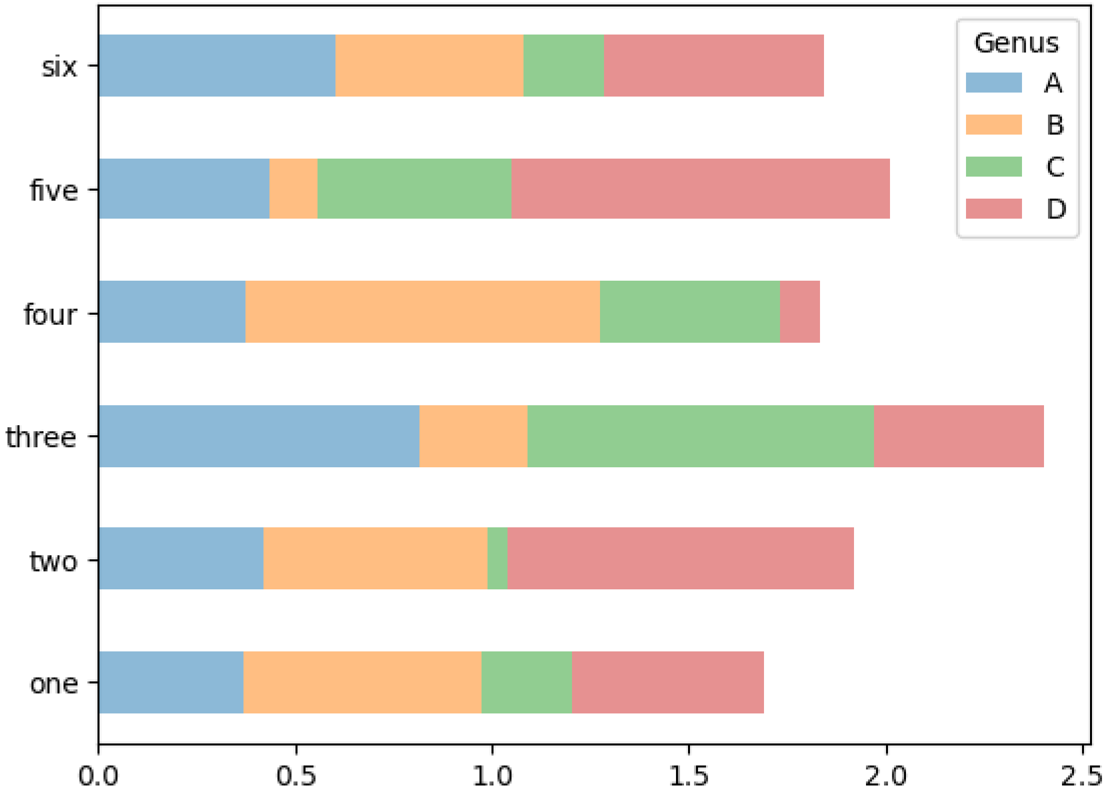
Out[14]:

```
<AxesSubplot:>
```



In [16]:
```
df.plot.barh(stacked=True, alpha=0.5)
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
plt.close('all')
```

In [20]:
```
tips = pd.read_csv('examples/tips.csv')
party_counts = pd.crosstab(tips['day'], tips['size'])
party_counts
```

Out[20]:

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

In [21]:

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

In [22]:

```
# Normalize to sum to 1
party_pcts = party_counts.div(party_counts.sum(1), axis=0)
party_pcts
```

Out[22]:

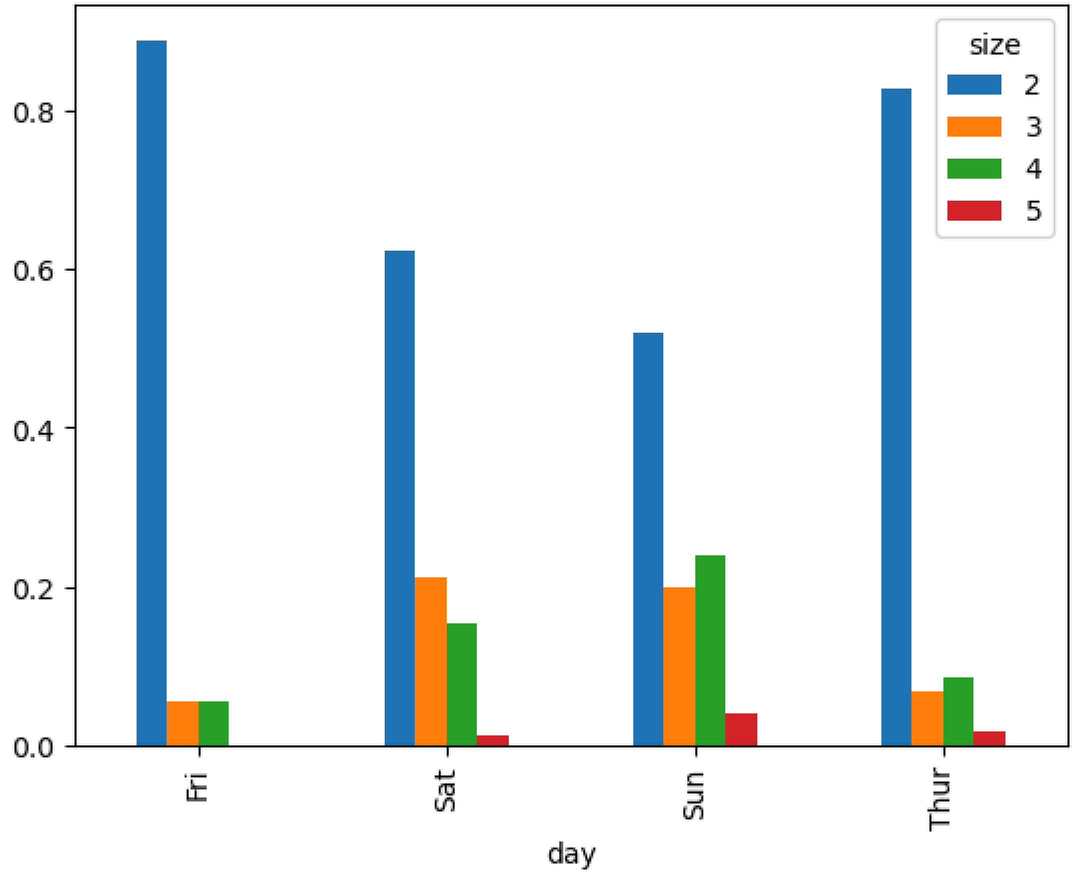
size	2	3	4	5
day				
Fri	0.888889	0.055556	0.055556	0.000000
Sat	0.623529	0.211765	0.152941	0.011765
Sun	0.520000	0.200000	0.240000	0.040000
Thur	0.827586	0.068966	0.086207	0.017241

In [23]:

```
party_pcts.plot.bar()
```

Out[23]:

```
<AxesSubplot:xlabel='day'>
```



In [24]:

```
plt.close('all')
```

In [26]:

```
import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
tips.head()
```

Out[26]:

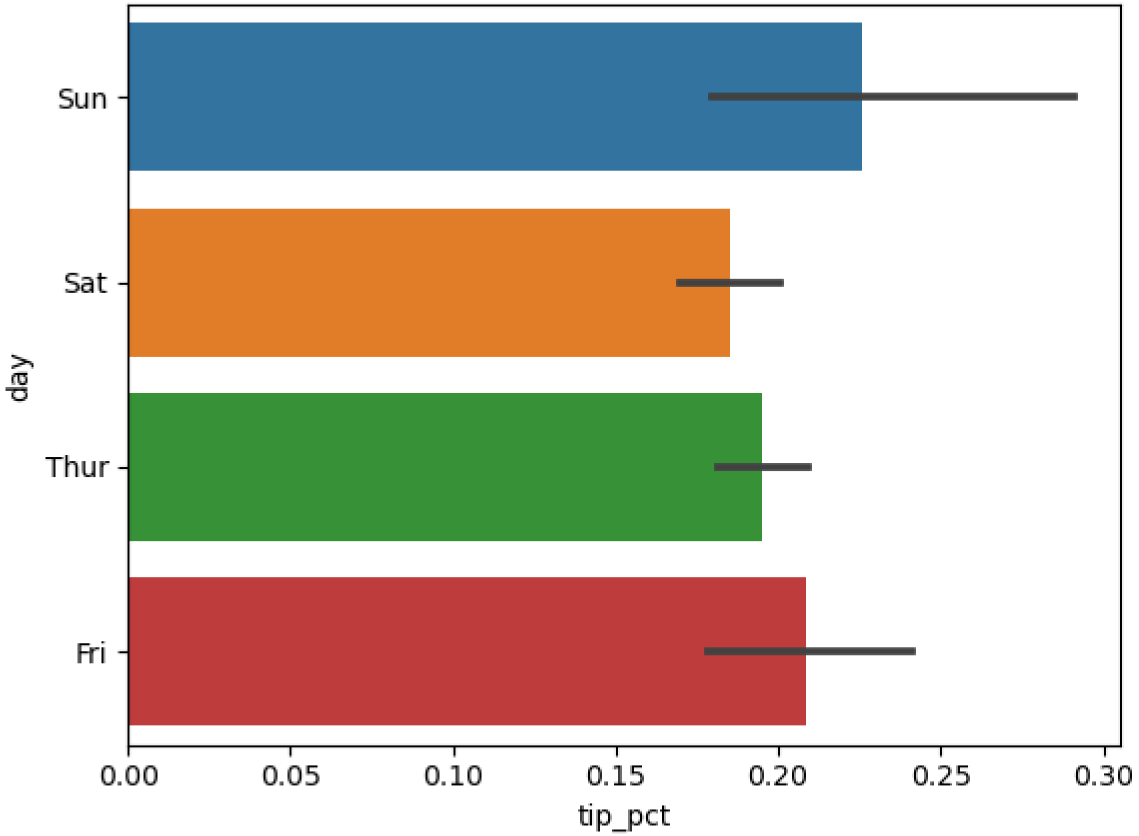
	total_bill	tip	smoker	day	time	size	tip_pct
0	16.99	1.01	No	Sun	Dinner	2	0.063204
1	10.34	1.66	No	Sun	Dinner	3	0.191244
2	21.01	3.50	No	Sun	Dinner	3	0.199886
3	23.68	3.31	No	Sun	Dinner	2	0.162494
4	24.59	3.61	No	Sun	Dinner	4	0.172069

In [25]:

```
import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
tips.head()
sns.barplot(x='tip_pct', y='day', data=tips, orient='h')
```

Out[25]:

```
<AxesSubplot:xlabel='tip_pct', ylabel='day'>
```



In [27]:

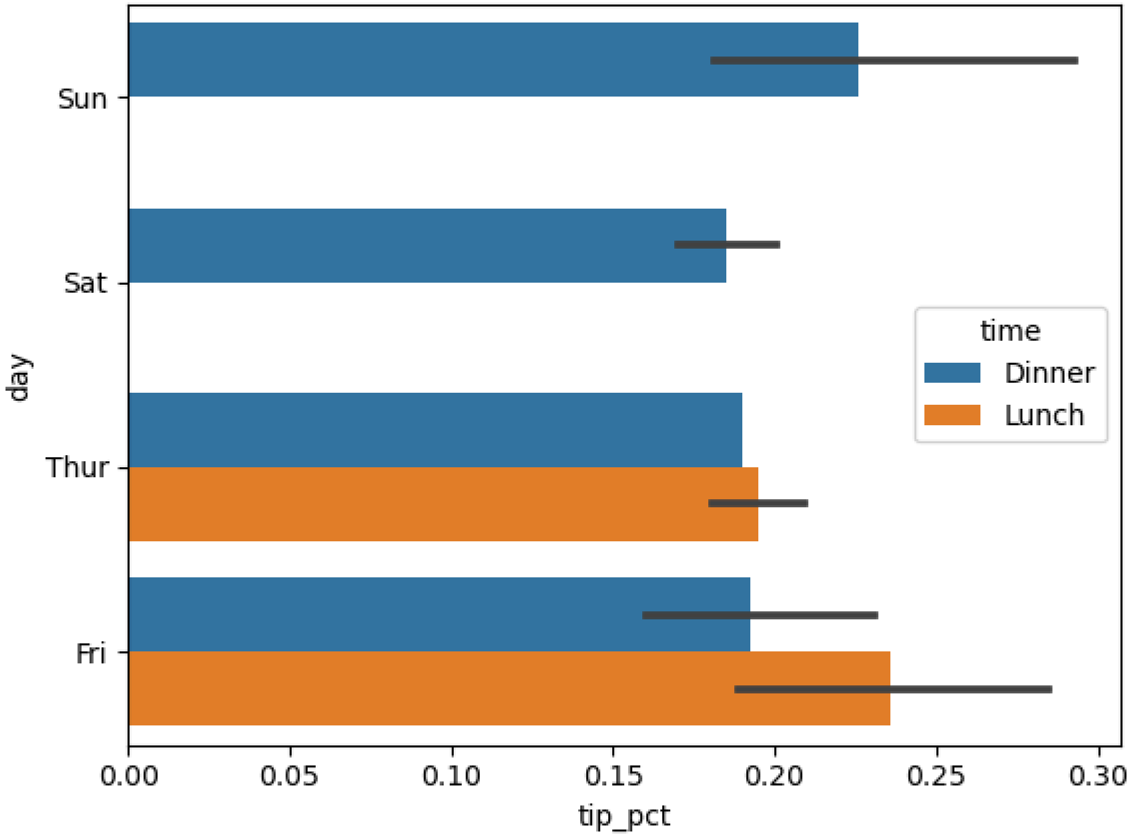
```
plt.close('all')
```

In [28]:

```
sns.barplot(x='tip_pct', y='day', hue='time', data=tips, orient='h')
```

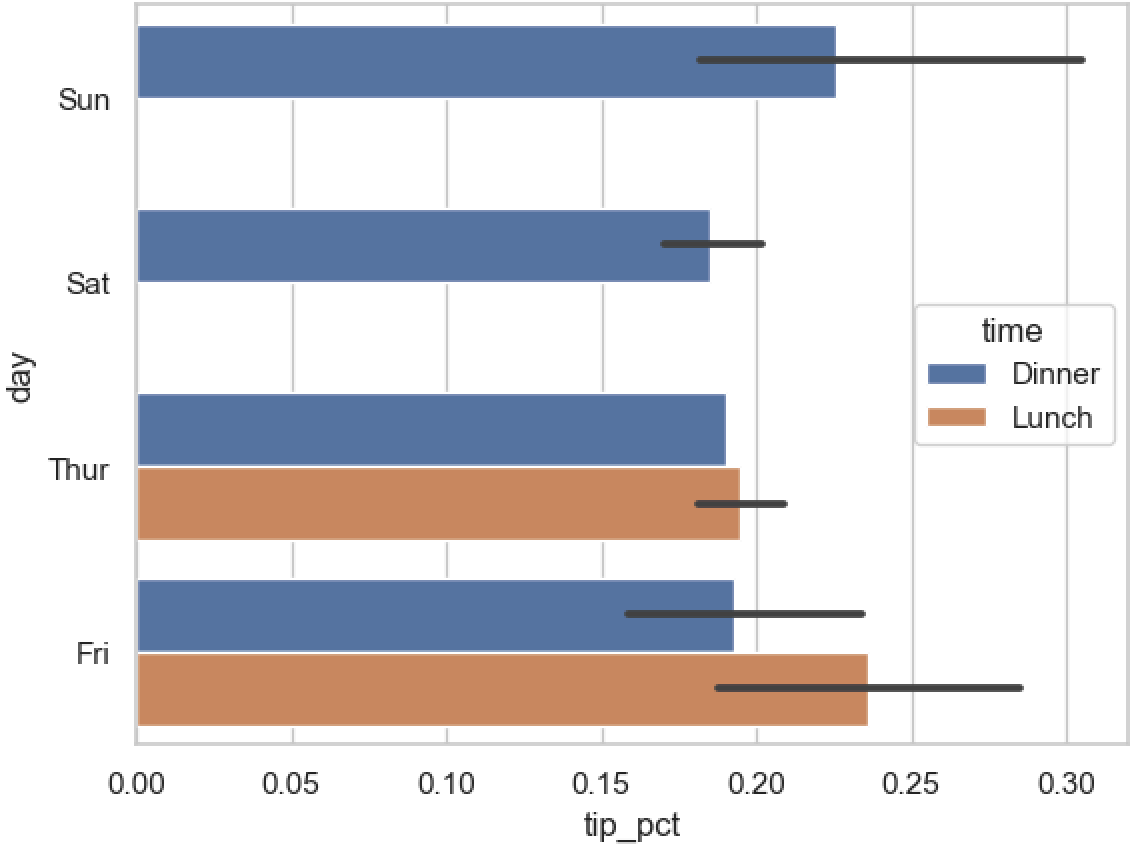
Out[28]:

```
<AxesSubplot:xlabel='tip_pct', ylabel='day'>
```



In [29]:
```
plt.close('all')
```

In [31]:
```
sns.barplot(x='tip_pct', y='day', hue='time', data=tips, orient='h')
sns.set(style="whitegrid")
```

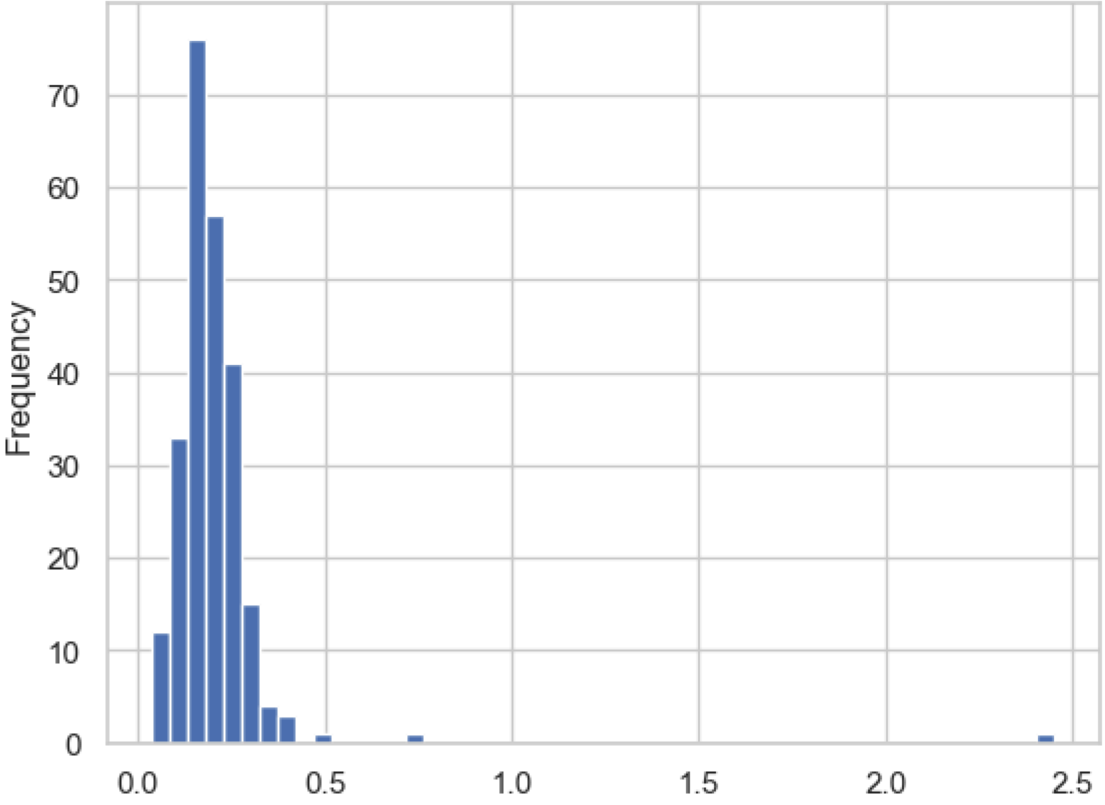


Histograms and Density Plots

In [ ]:
```
plt.figure()
```

In [32]:
```
tips['tip_pct'].plot.hist(bins=50)
```

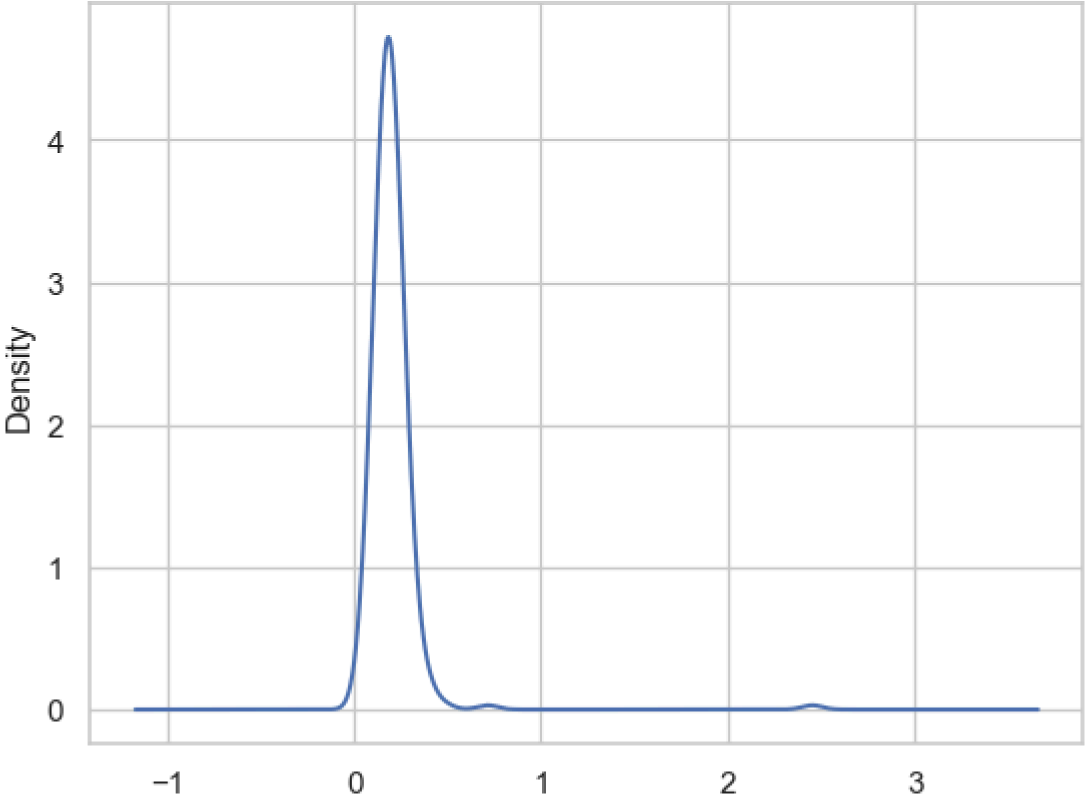
Out[32]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [ ]:
```
plt.figure()
```

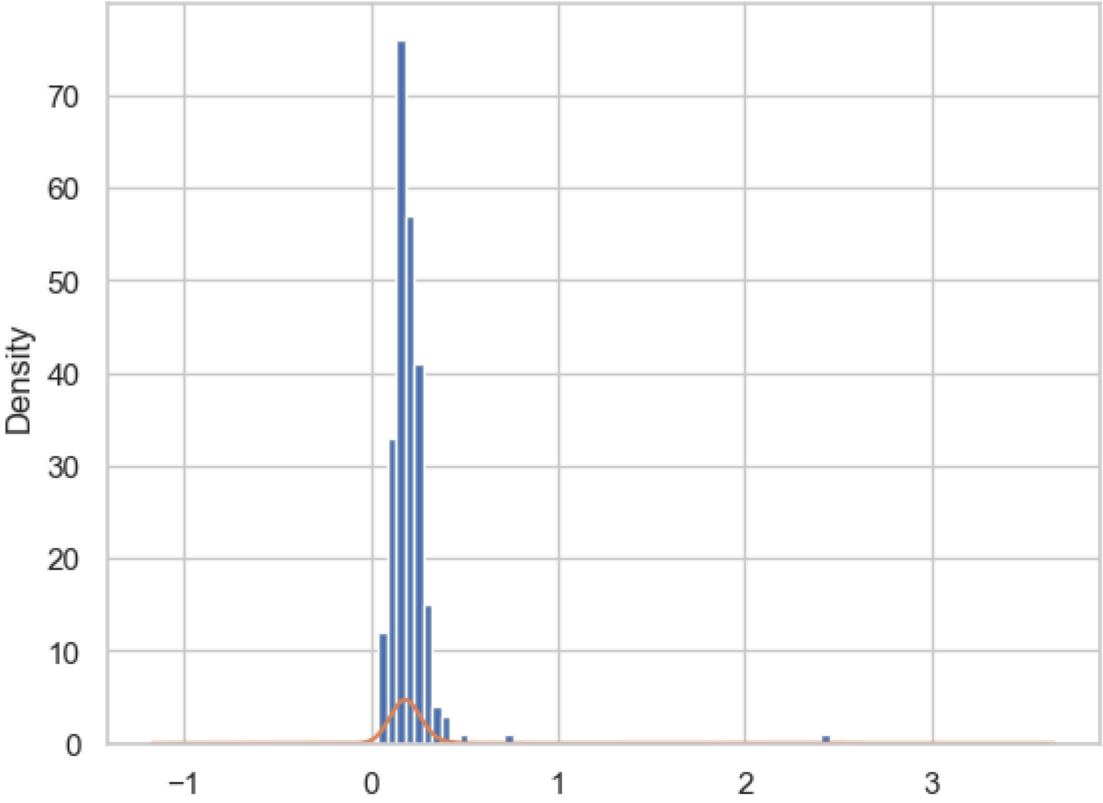
In [33]:
```
tips['tip_pct'].plot.density()
```

Out[33]:
```
<AxesSubplot:ylabel='Density'>
```



```
In [39]: plt.figure()
         tips['tip_pct'].plot.hist(bins=50)
         tips['tip_pct'].plot.density()
```

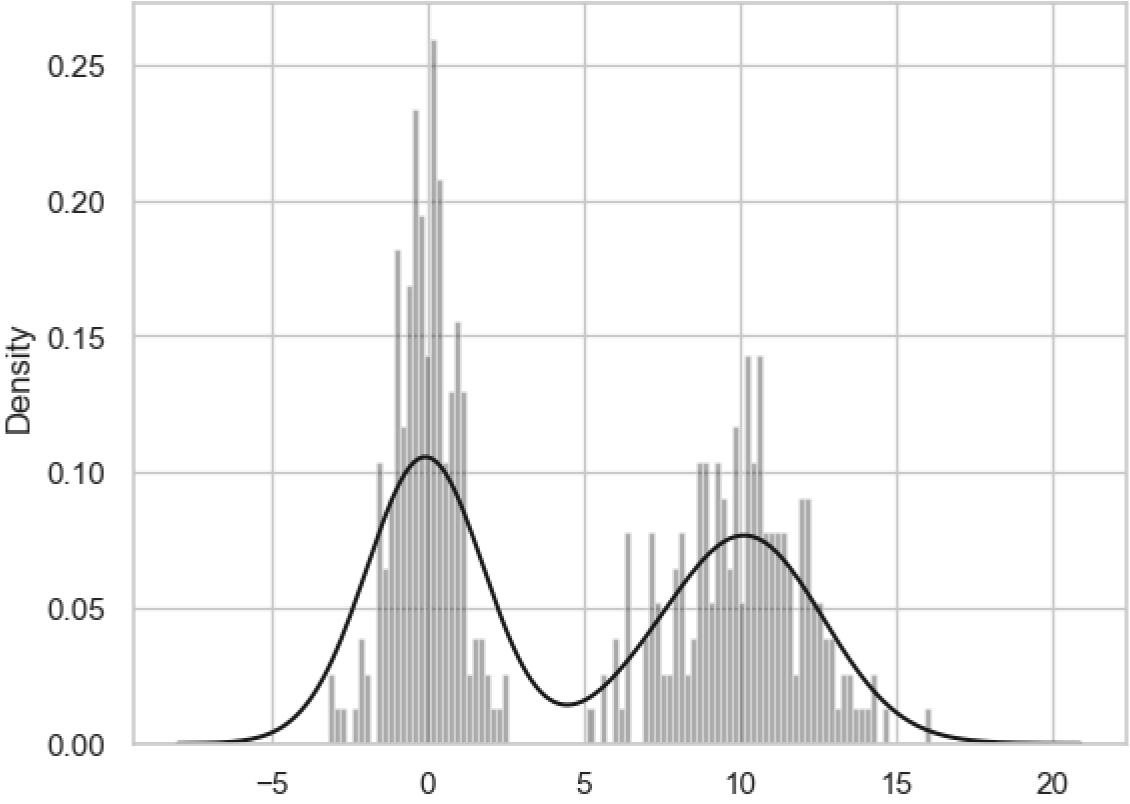
```
Out[39]: <AxesSubplot:ylabel='Density'>
```



```
In [40]: 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')
```

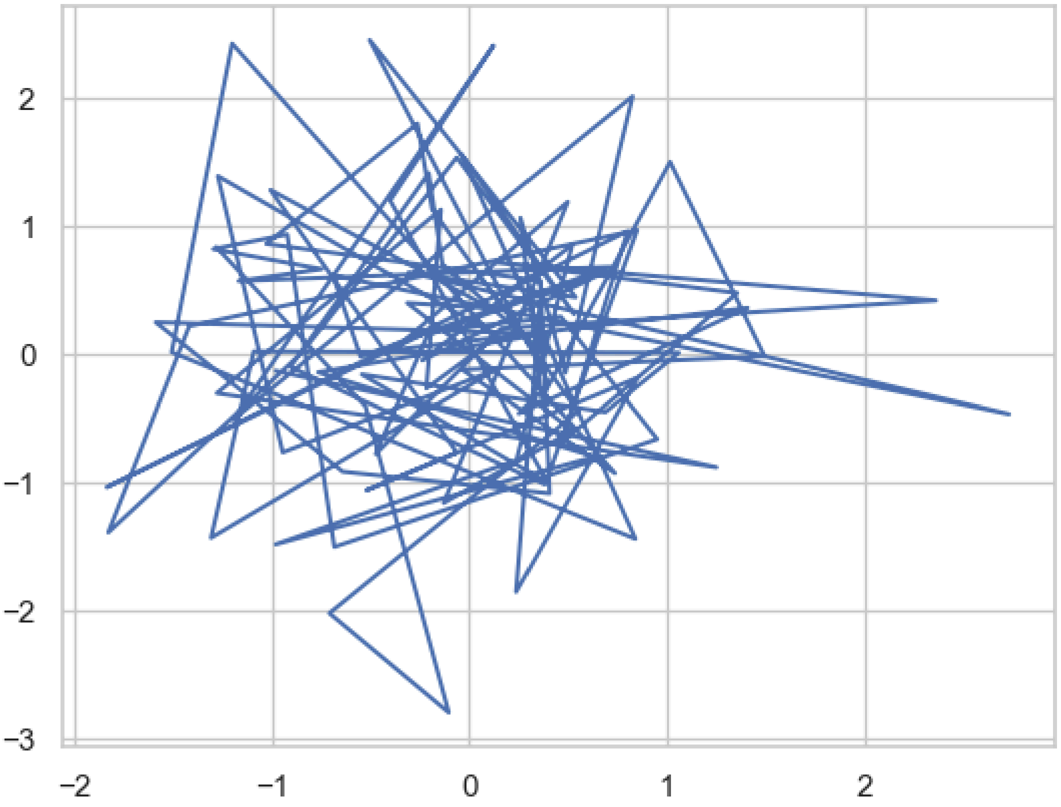
```
C:\PythonDSA\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with simil
ar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[40]: <AxesSubplot:ylabel='Density'>
```



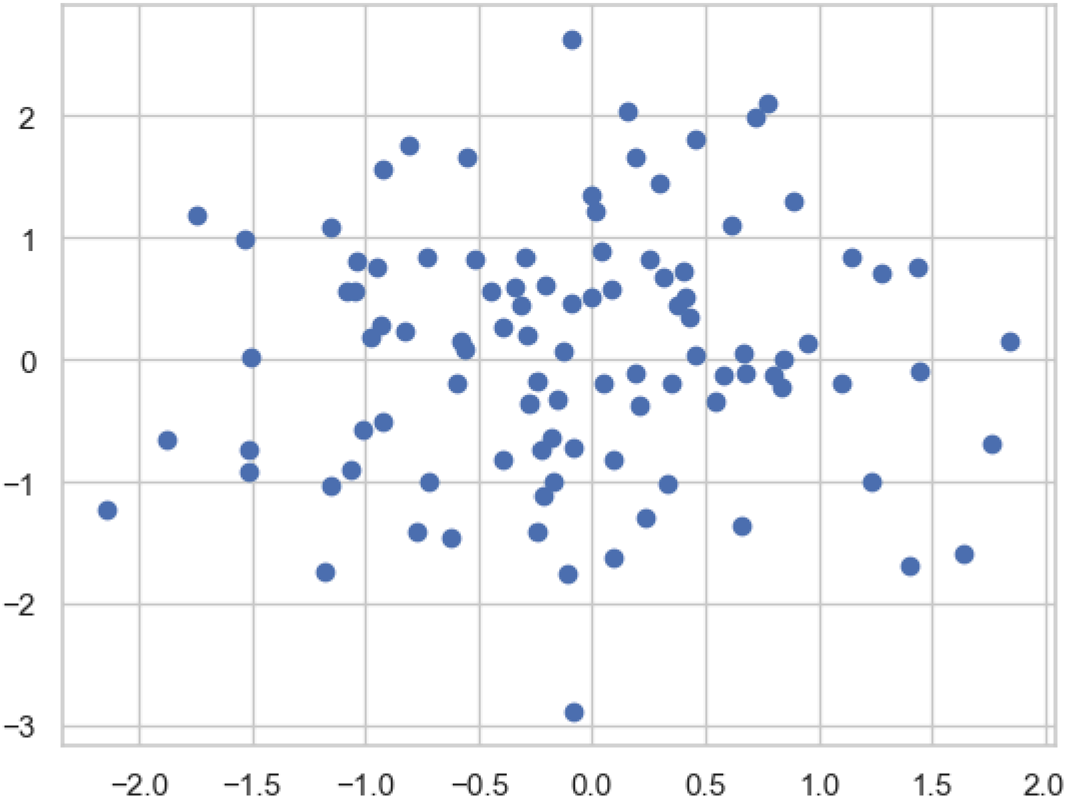
```
In [41]: s = pd.Series(np.random.randn(100), index=np.random.randn(100))
         s.plot()
```

```
Out[41]: <AxesSubplot:>
```



In [42]:
```
s = pd.Series(np.random.randn(100), index=np.random.randn(100))
s.plot(style='o', linestyle='')
```

Out[42]:
```
<AxesSubplot:>
```



Scatter or Point Plots

In [59]:
```
macro = pd.read_csv('examples/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
```

```
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
```

Out[59]:

	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

In [ ]:

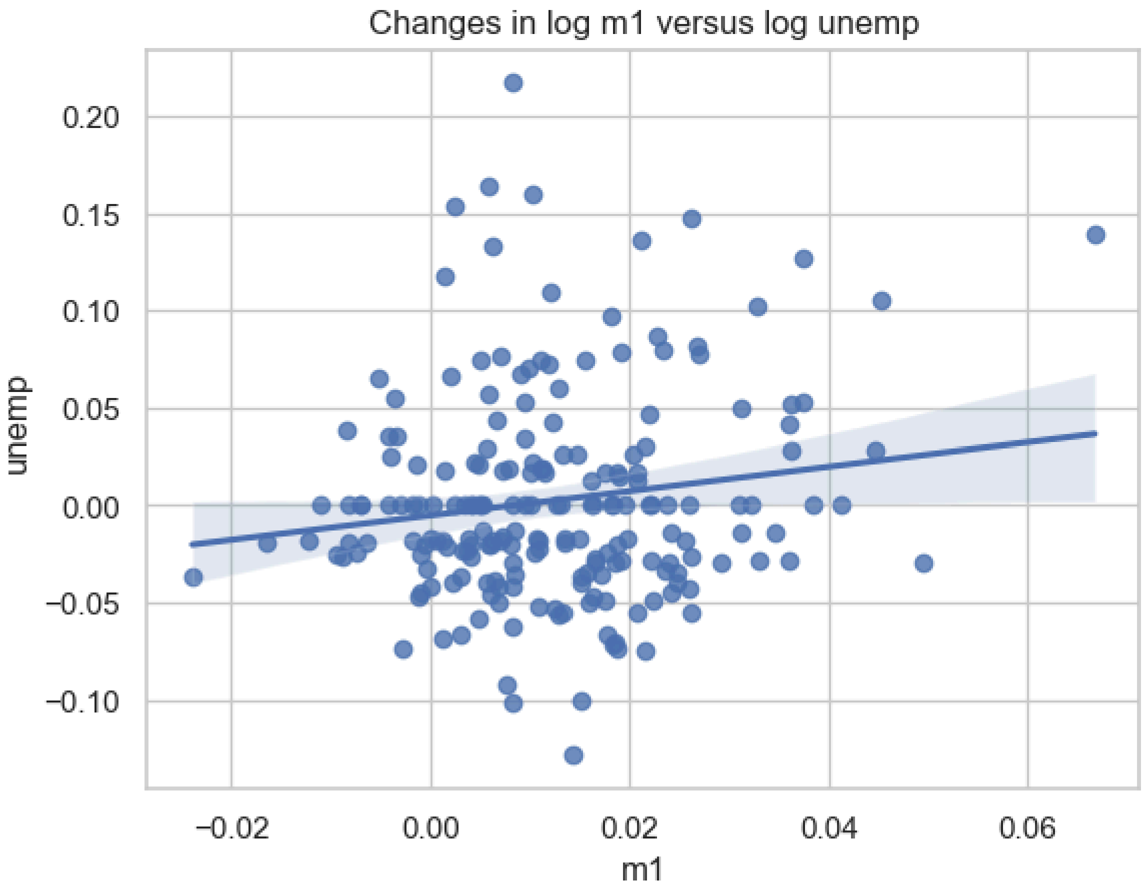
```
plt.figure()
```

In [62]:

```
sns.regplot(x='m1', y='unemp', data=trans_data)
plt.title('Changes in log %s versus log %s' % ('m1', 'unemp'))
```

Out[62]:

```
Text(0.5, 1.0, 'Changes in log m1 versus log unemp')
```

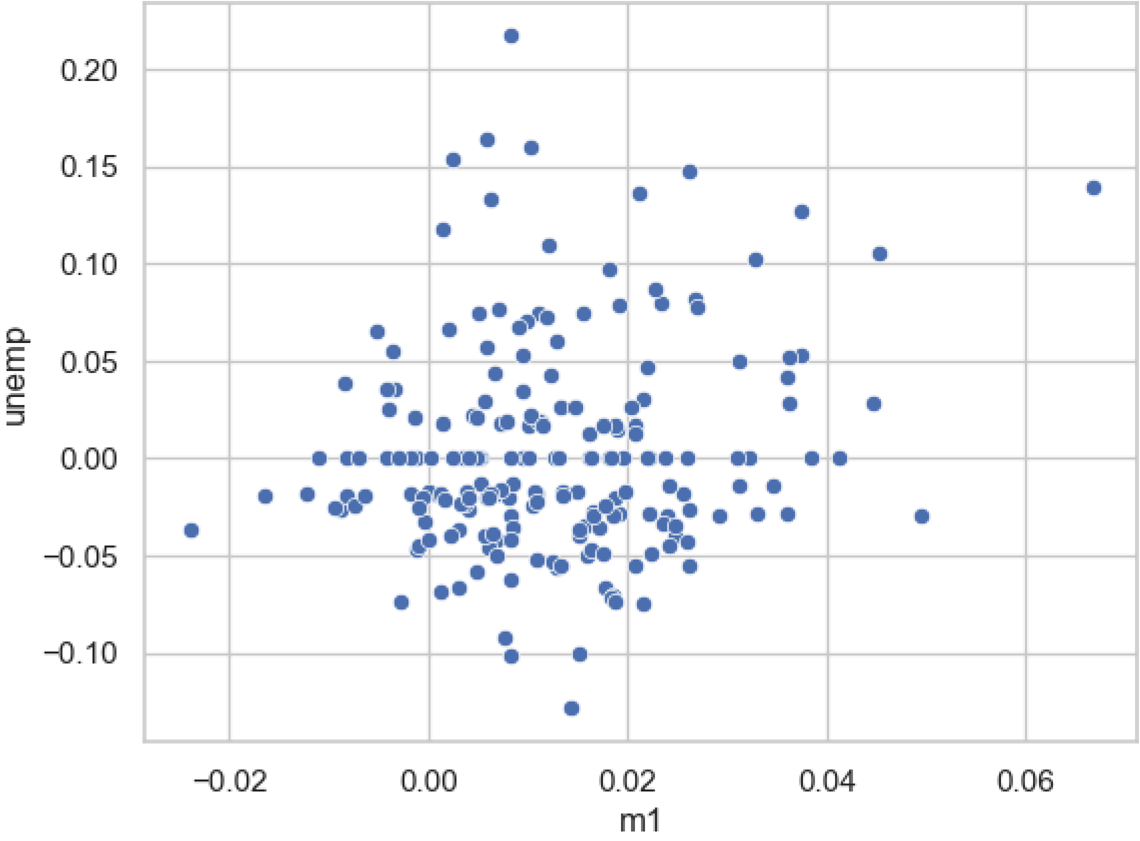


In [61]:

```
sns.scatterplot(x='m1', y='unemp', data=trans_data)
```

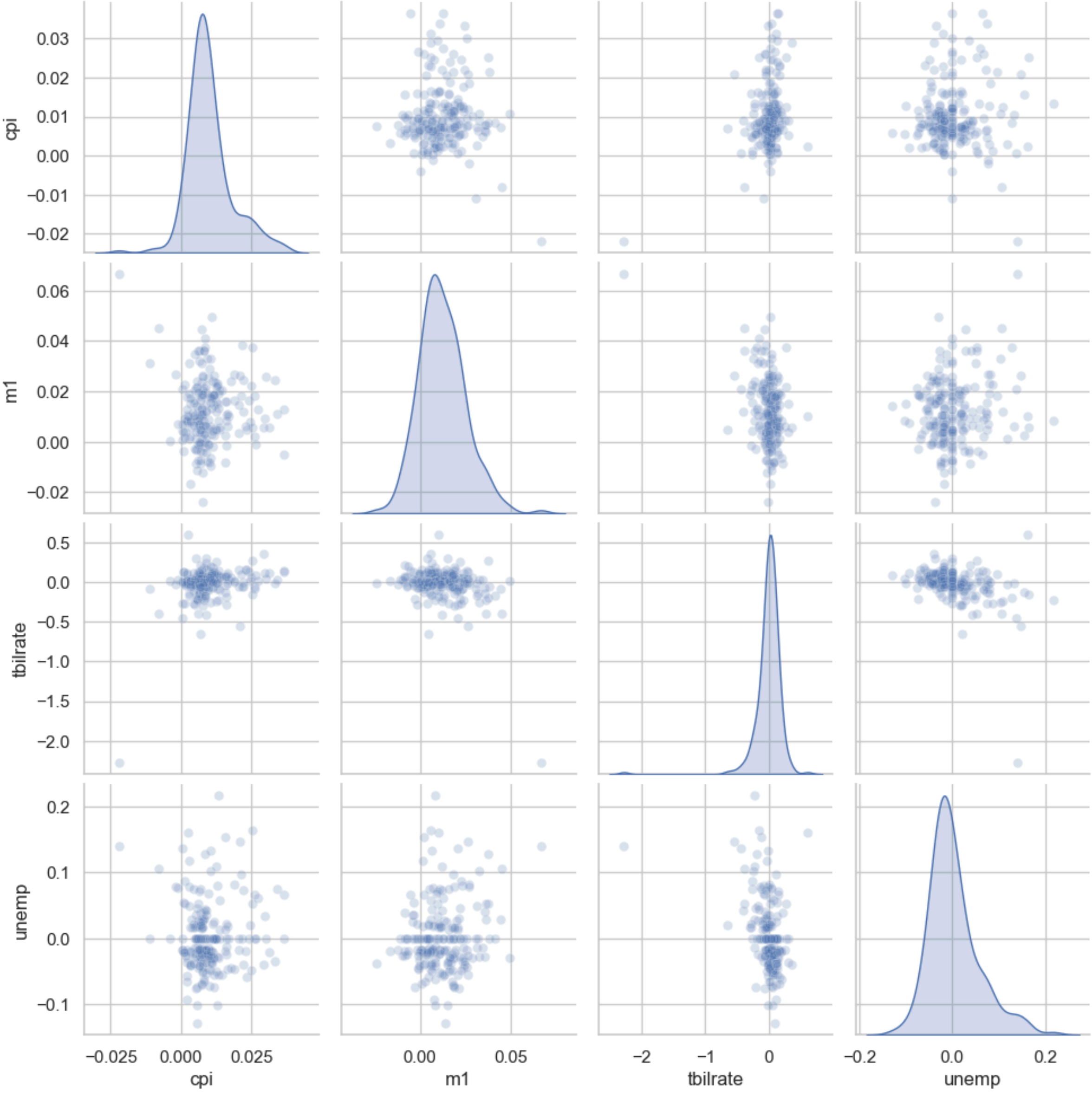
Out[61]:

```
<AxesSubplot:xlabel='m1', ylabel='unemp'>
```



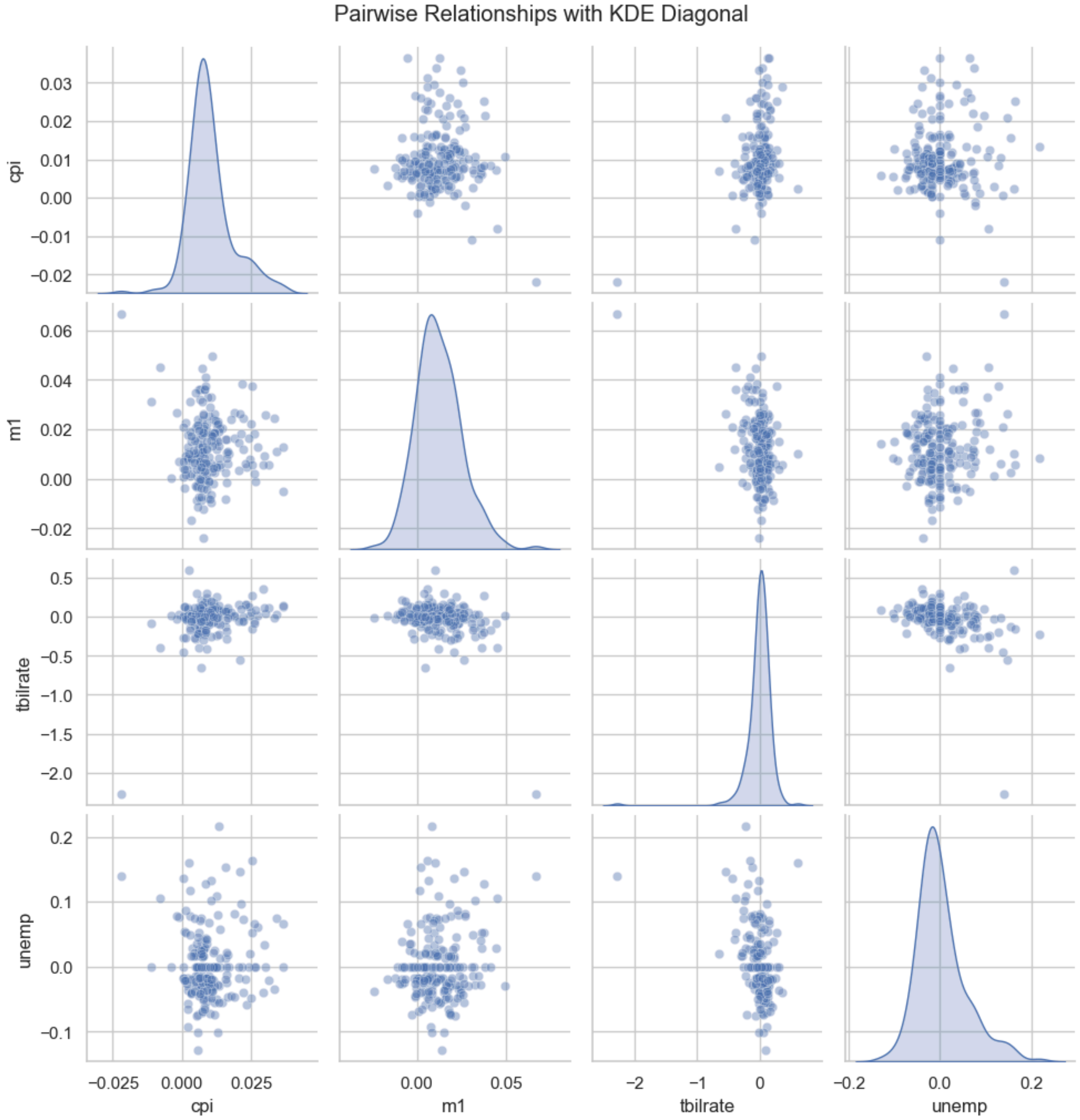
In [63]:
```
sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.2})
```

Out[63]:
```
<seaborn.axisgrid.PairGrid at 0x1bde8407f70>
```



In [65]:

```
sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.4})
plt.suptitle('Pairwise Relationships with KDE Diagonal', y=1.02)  # Adjust the titl
plt.show()
```



Facet Grids and Categorical Data

In [67]:
```
import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
tips
```

Out[67]:

	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
...	...	...	...	...	...	...	...
239	29.03	5.92	No	Sat	Dinner	3	0.256166
240	27.18	2.00	Yes	Sat	Dinner	2	0.079428
241	22.67	2.00	Yes	Sat	Dinner	2	0.096759
242	17.82	1.75	No	Sat	Dinner	2	0.108899
243	18.78	3.00	No	Thur	Dinner	2	0.190114

244 rows × 7 columns

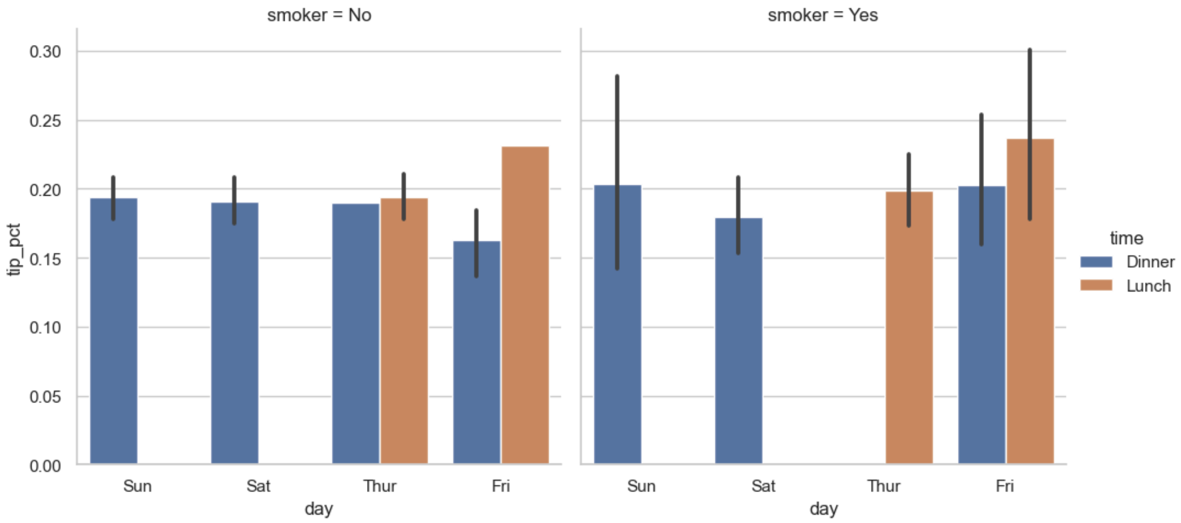
In [68]:

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

```
C:\PythonDSA\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning:
The `factorplot` function has been renamed to `catplot`. The original name will be
removed in a future release. Please update your code. Note that the default `kind`
in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[68]:

```
<seaborn.axisgrid.FacetGrid at 0x1bdeac761c0>
```

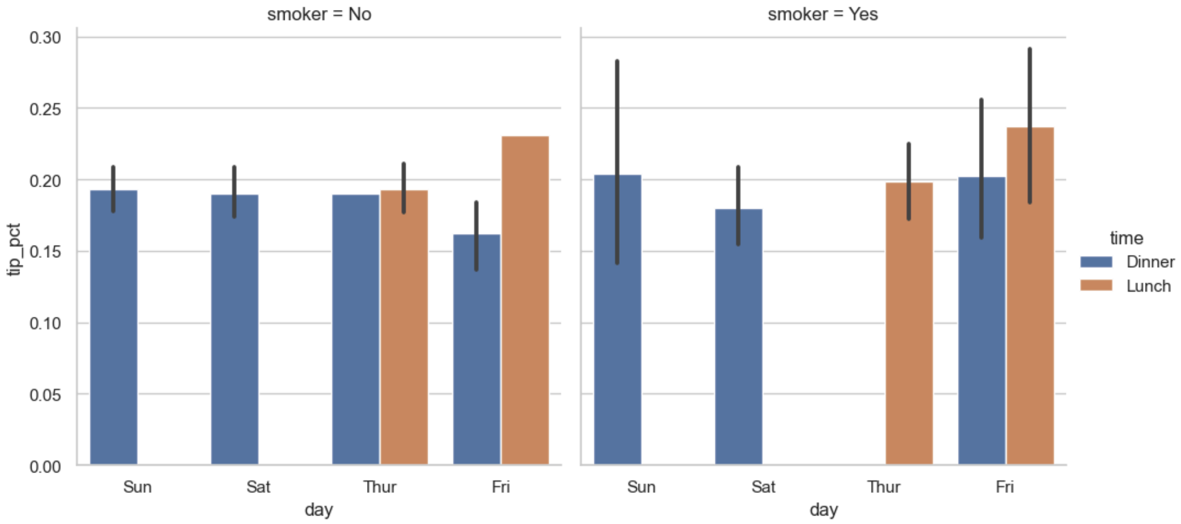


In [69]:

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

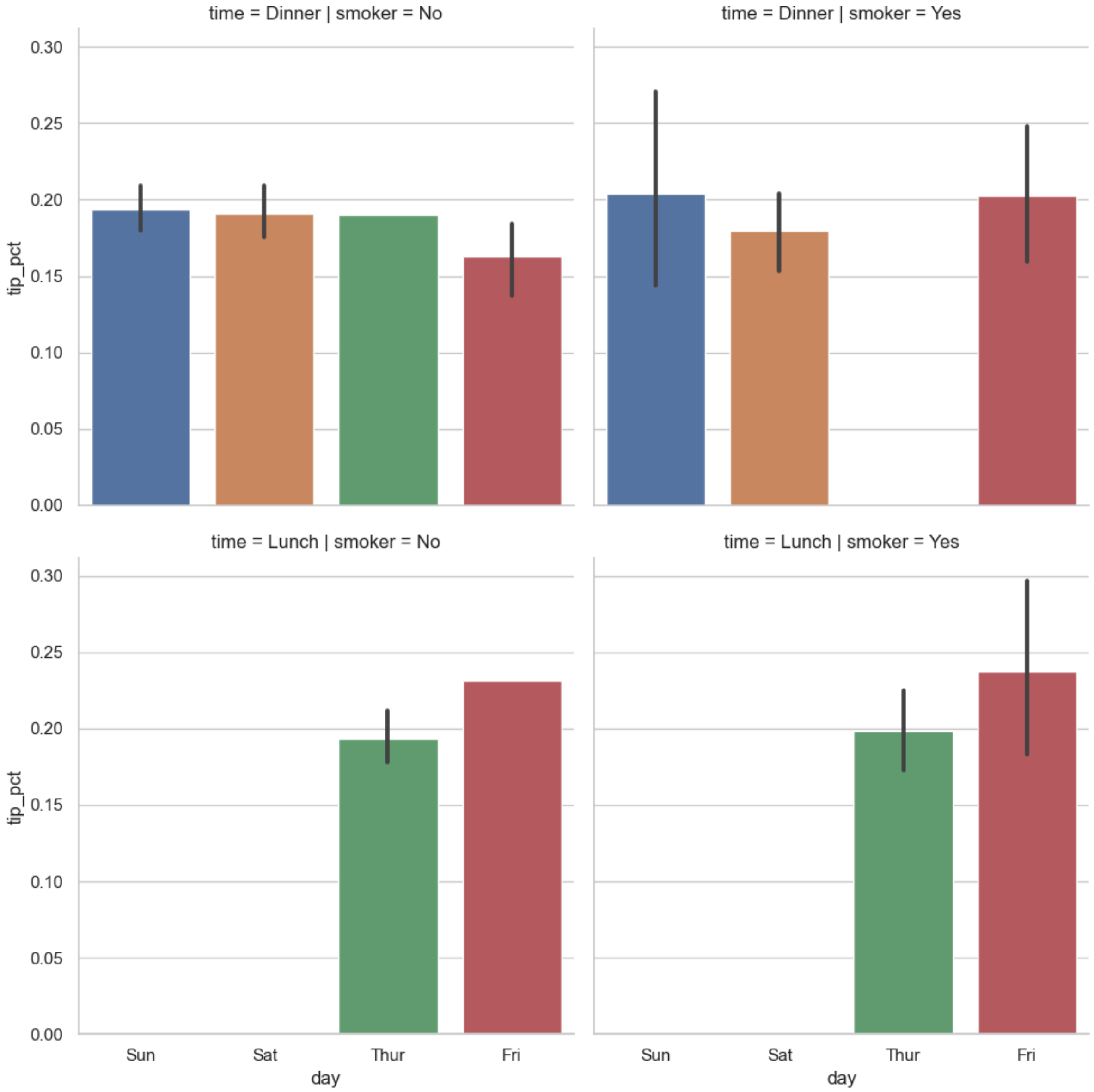
Out[69]:

```
<seaborn.axisgrid.FacetGrid at 0x1bdeba33970>
```



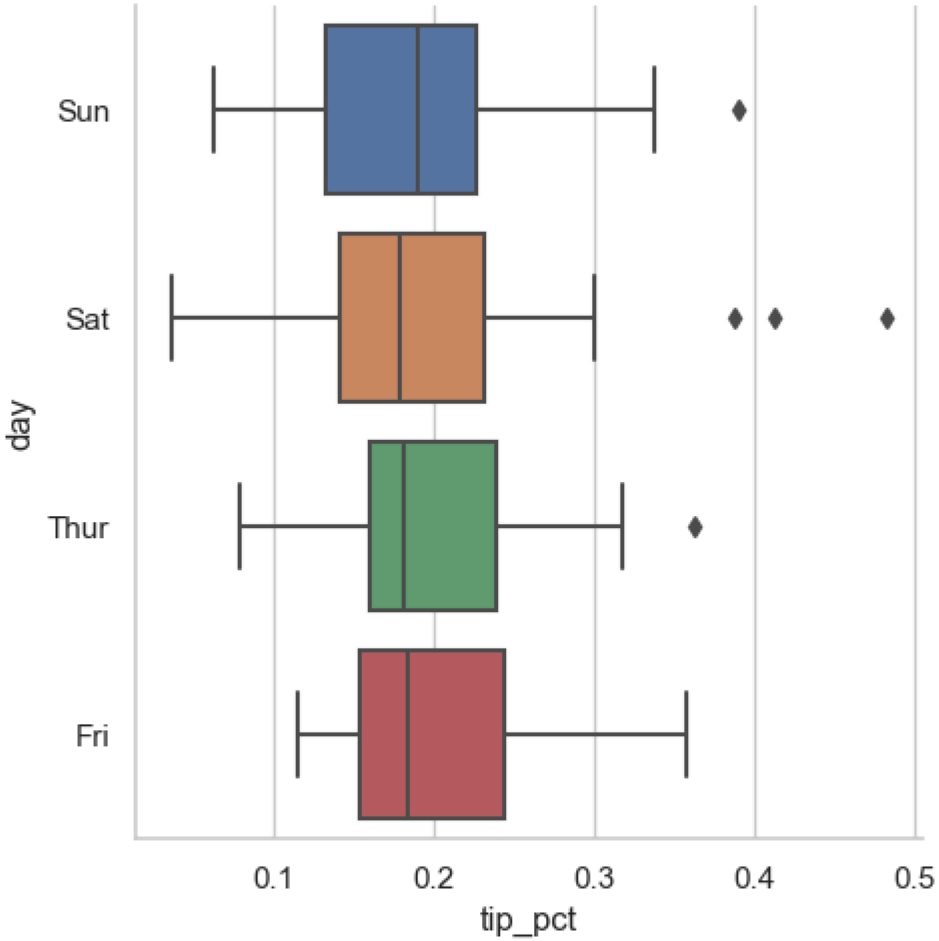
In [71]:
```
sns.catplot(x='day', y='tip_pct', row='time',
               col='smoker',
               kind='bar', data=tips[tips.tip_pct < 1])
```

Out[71]:
```
<seaborn.axisgrid.FacetGrid at 0x1bdec2a8640>
```



In [73]:
```
sns.catplot(x='tip_pct', y='day', kind='box',
               data=tips[tips.tip_pct < 0.5])
```

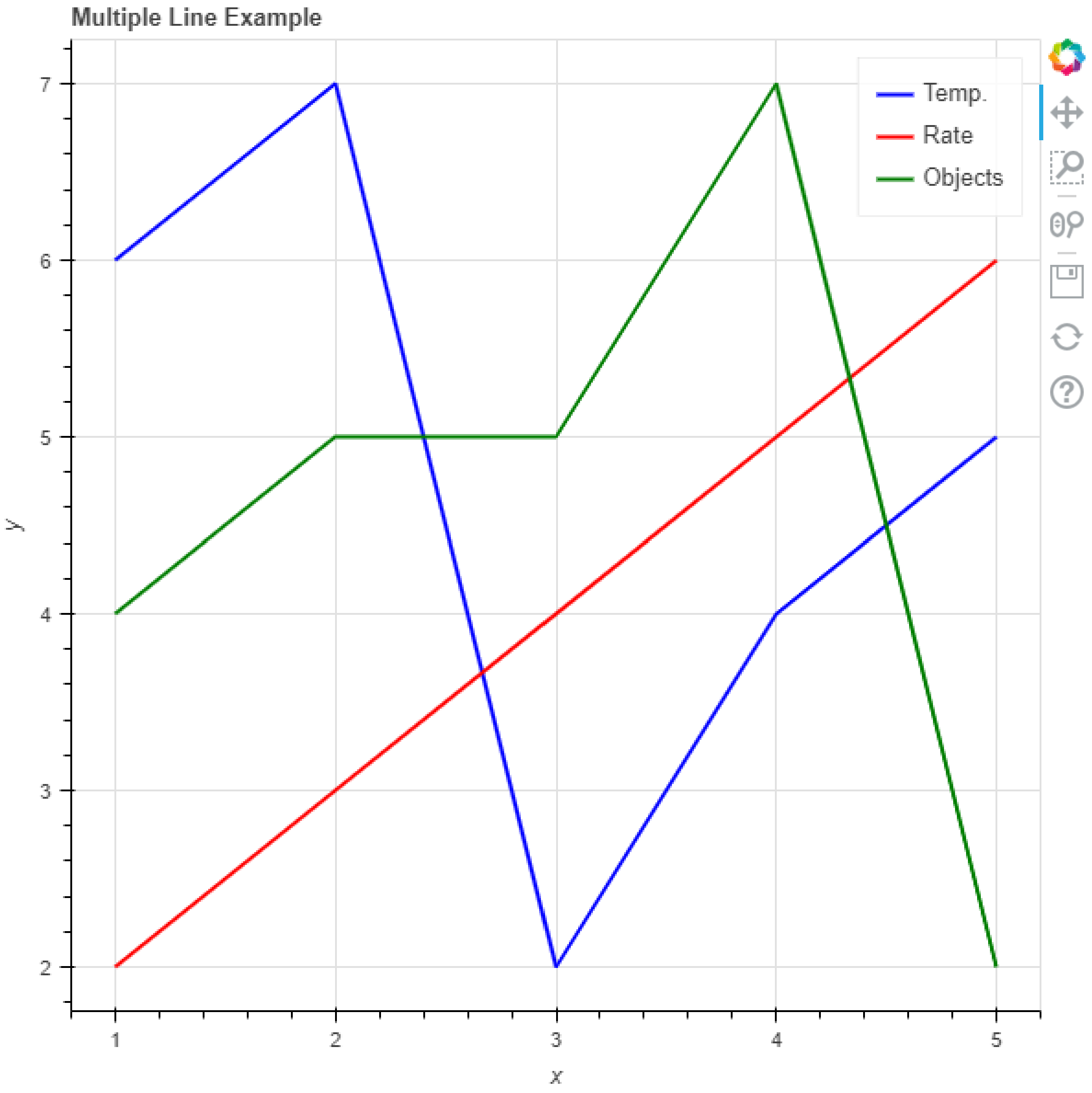
Out[73]:
```
<seaborn.axisgrid.FacetGrid at 0x1bdeb4cccd0>
```



Other Python Visualization Tools

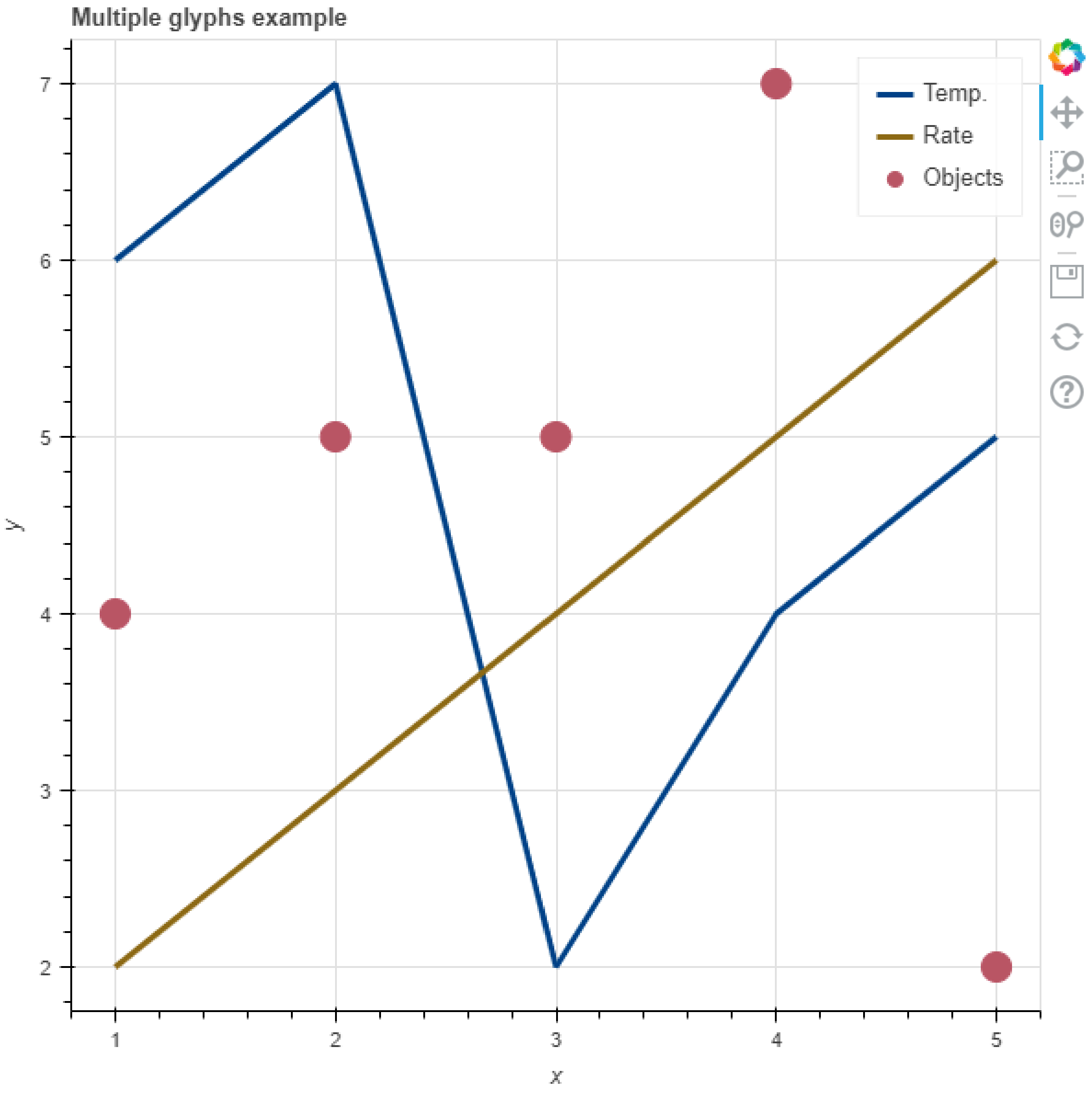
In [78]:
```
from bokeh.plotting import figure, show, output_notebook

# Prepare some data
x = [1, 2, 3, 4, 5]
y1 = [6, 7, 2, 4, 5]
y2 = [2, 3, 4, 5, 6]
y3 = [4, 5, 5, 7, 2]

# Set up Bokeh to display plots inline in the notebook
output_notebook()

# Create a new plot with a title and axis labels
p = figure(title="Multiple Line Example", x_axis_label="x", y_axis_label="y")

# Add multiple renderers
p.line(x, y1, legend_label="Temp.", color="blue", line_width=2)
p.line(x, y2, legend_label="Rate", color="red", line_width=2)
p.line(x, y3, legend_label="Objects", color="green", line_width=2)

# Show the results within the notebook
show(p)
```

Loading BokehJS ...



In [79]:
```
from bokeh.plotting import figure, show

# prepare some data
x = [1, 2, 3, 4, 5]
y1 = [6, 7, 2, 4, 5]
y2 = [2, 3, 4, 5, 6]
y3 = [4, 5, 5, 7, 2]

output_notebook()
# create a new plot with a title and axis labels
p = figure(title="Multiple glyphs example", x_axis_label="x", y_axis_label="y")

# add multiple renderers
p.line(x, y1, legend_label="Temp.", color="#004488", line_width=3)
p.line(x, y2, legend_label="Rate", color="#906c18", line_width=3)
p.scatter(x, y3, legend_label="Objects", color="#bb5566", size=16)

# show the results
show(p)
```

BokehJS 2.4.3 successfully loaded.



Conclusion