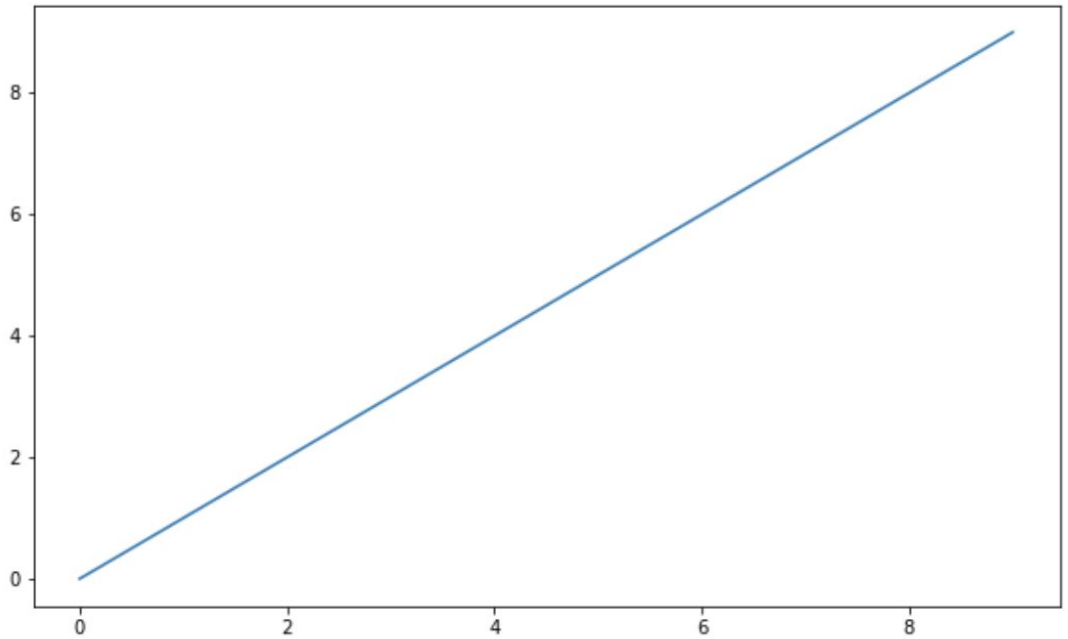
UVA | SCHOOL of DATA SCIENCE

Part I: A Brief matplotlib API Primer

Ex. #1 Simple data plot

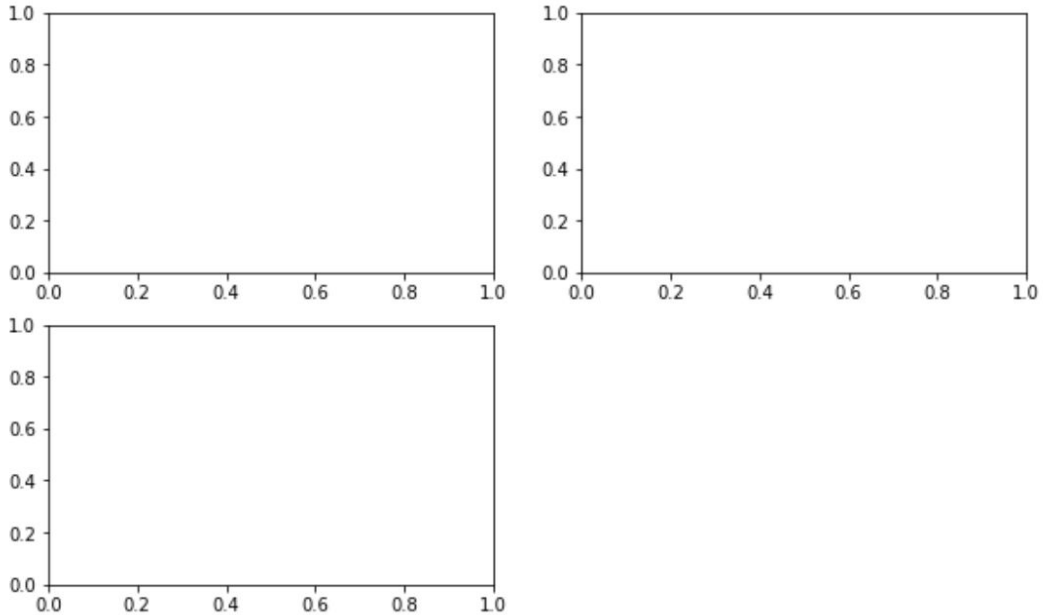
```
import numpy as np
data. np.arange(10)
data
plt.plot(data)
```

```
[<matplotlib.lines.Line2D at 0x115895fd0>]
```



Ex. #2 Figures and Subplots in a given figure

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

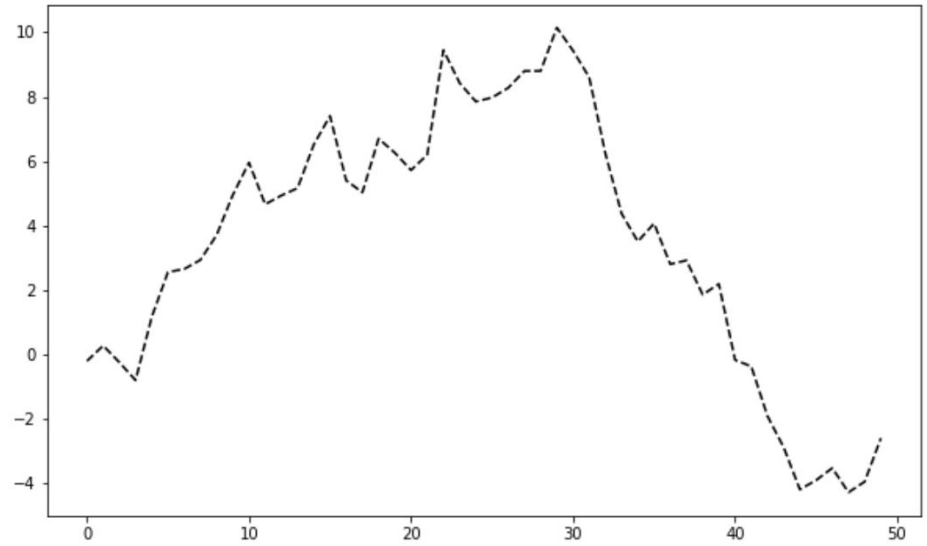


- You can specify the overall configuration of the sub plots (e.g. 2x2)
- Also, you can specify which location you want certain plots (e.g. third parameter)

Ex. #3 creating a specific plot; also specifying the kind of line you want (black; dashed)

```
plt.plot(np.random.randn(50).cumsum(), 'k--')
```

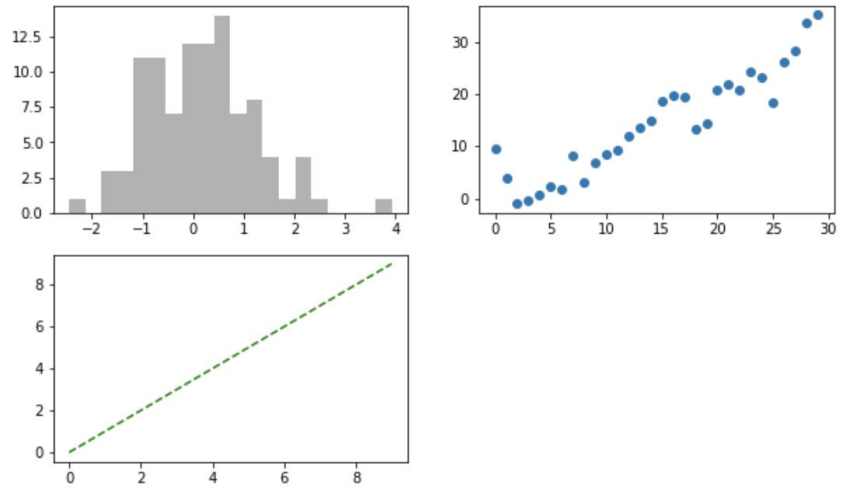
```
[<matplotlib.lines.Line2D at 0x115aeddd8>]
```



Ex. #4 Putting it all together

```
fig = plt.figure()

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

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))
ax3.plot(np.arange(10), 'g--') #green dashed line
```

```
[<matplotlib.lines.Line2D at 0x115fab588>]
```

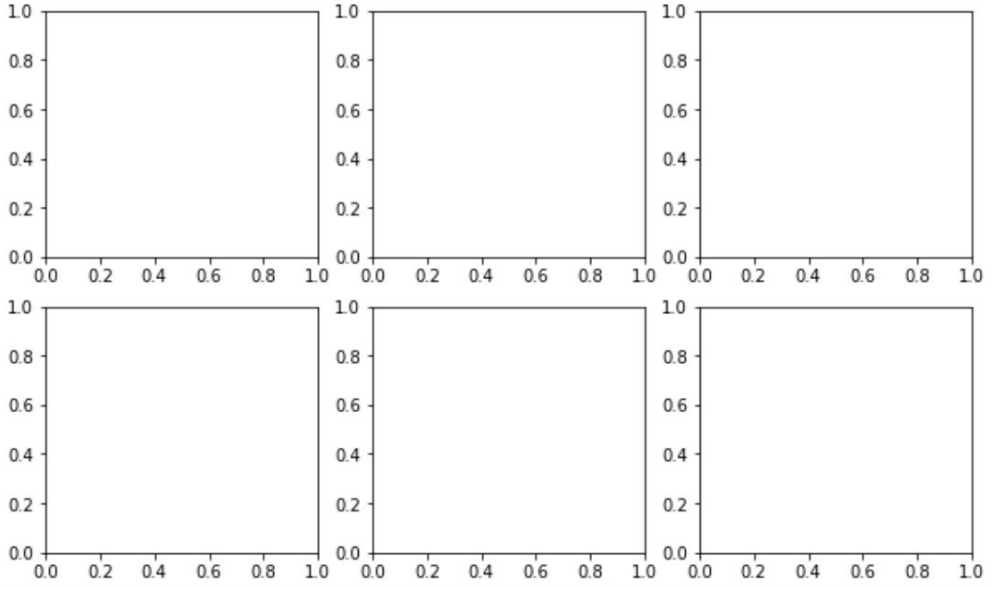


- Specify the kind of plots you want by utilizing the plot variables ax1, ax2 and ax3
- First is a histogram
- Second is a scatter plot
- Third is a simple lot (notice choice of green dashed line)

Ex. #5 Close a figure window

```
plt.close('all')

fig, axes = plt.subplots(2,3)
axes
```

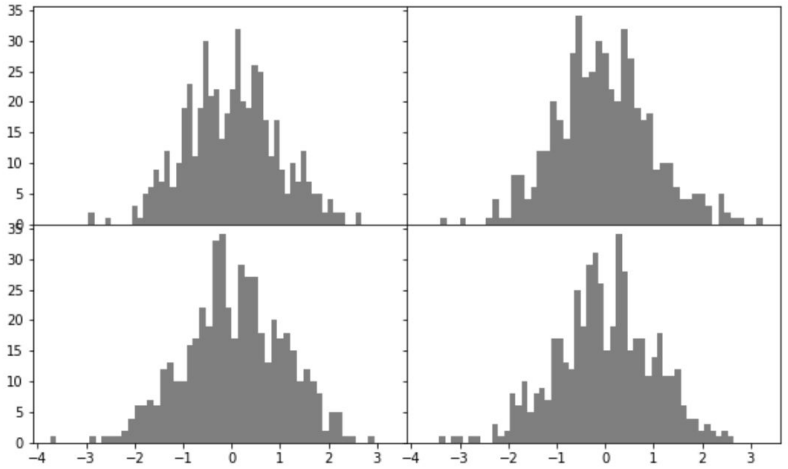
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11604df60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116151518>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116178a90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1161aa048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1161cf5c0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1161f8b38>]],
      dtype=object)
```



Ex. #6 Adjusting the spacing around subplots

```
Subplots_adjust(left=None, bottom=None, right=None, top=None, wspace =None, hspace=None)
```

```
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)
```



- Note plt.subplots returns two things: " fig " and “axes”
- You can specify which subplot you want by indexing into "axes": `axes[i, j]`
- Once the plots have been created then you can call " `subplots_adjust()`" function

UVA | SCHOOL of DATA SCIENCE

Part II:
Colors, Markers, and Line Styles

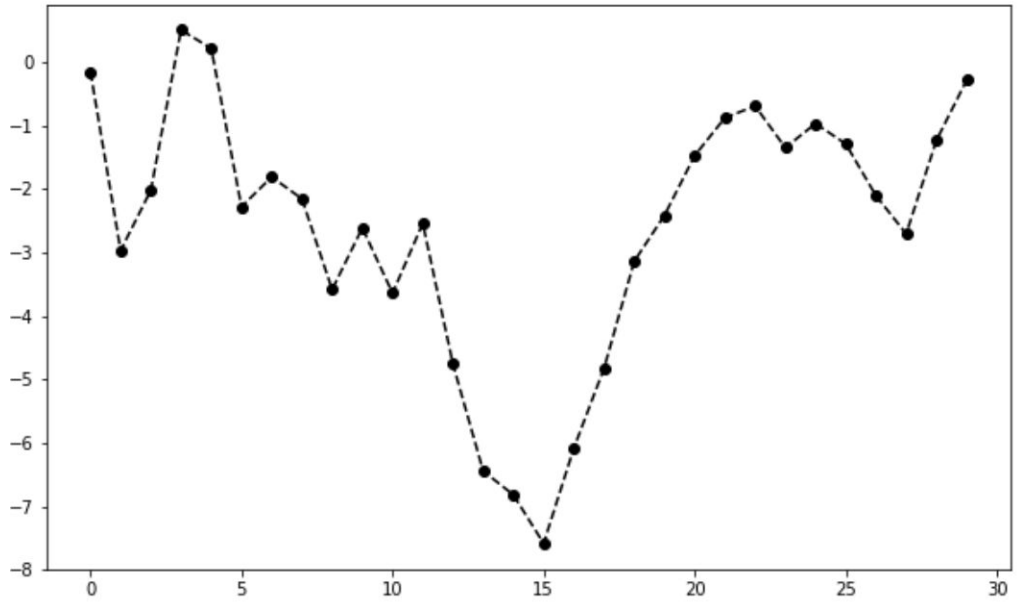
Ex. #7 Colors, Markers, and Line Styles

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

Note the second parameter:

- Color (e.g. k=black)
- Style of marker
- Style of line

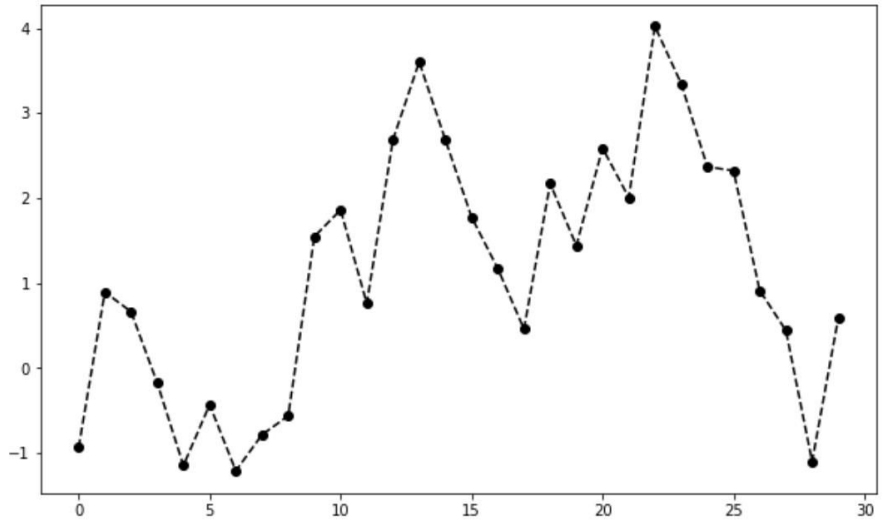
```
[<matplotlib.lines.Line2D at 0x1168ecd30>]
```



Ex. #8 Another way of plotting, with ' color','linestyle ', and 'marker' clearly

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

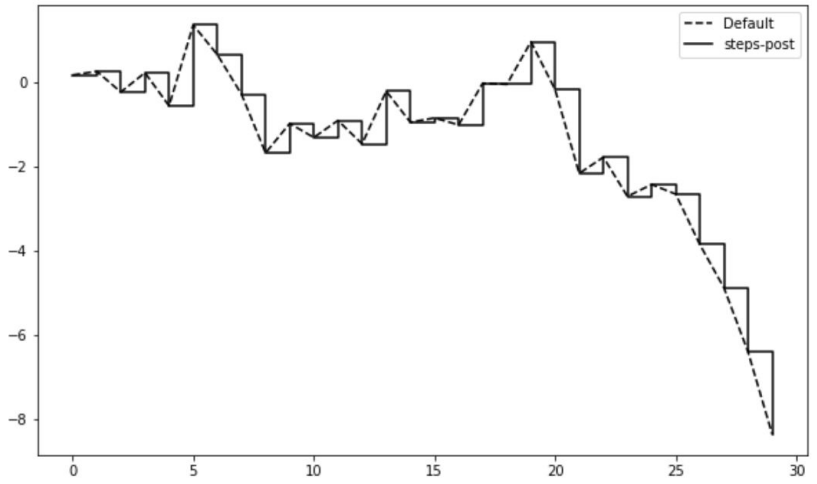
```
[<matplotlib.lines.Line2D at 0x116b5a8d0>]
```



Ex. #9 Steps-post

```
plt.close('all')

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')
```

```
<matplotlib.legend.Legend at 0x116d671d0>
```



- Multiple plots superimposed on one another
- Adding a legend here, too

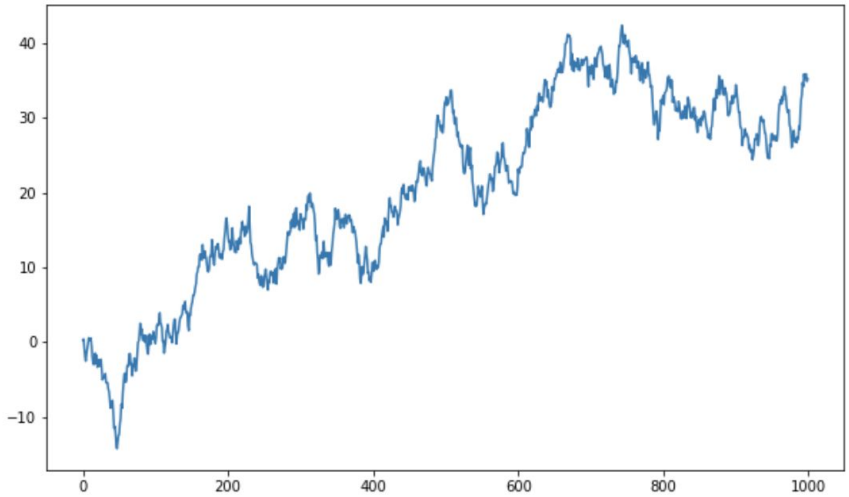
UVA | SCHOOL of DATA SCIENCE

Part III:
Ticks, Labels, and Legends

Ex. #10 Setting the title, axis labels , ticks , and ticklabels

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

```
[<matplotlib.lines.Line2D at 0x116dd17f0>]
```

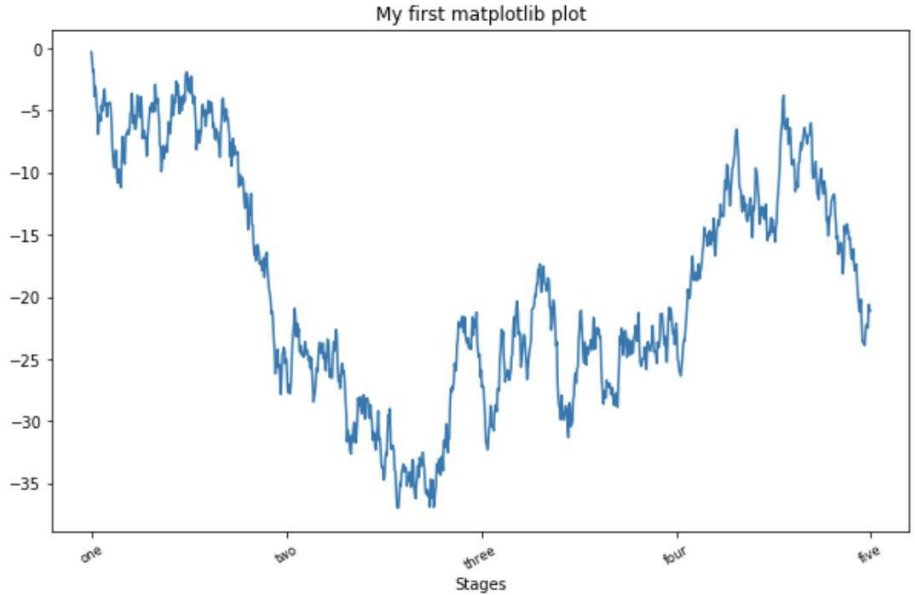


There's a lot we can add to this plot!

Ex. #11 Setting the title, axis labels , ticks , and ticklabels. Putting it all together

```
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')
```

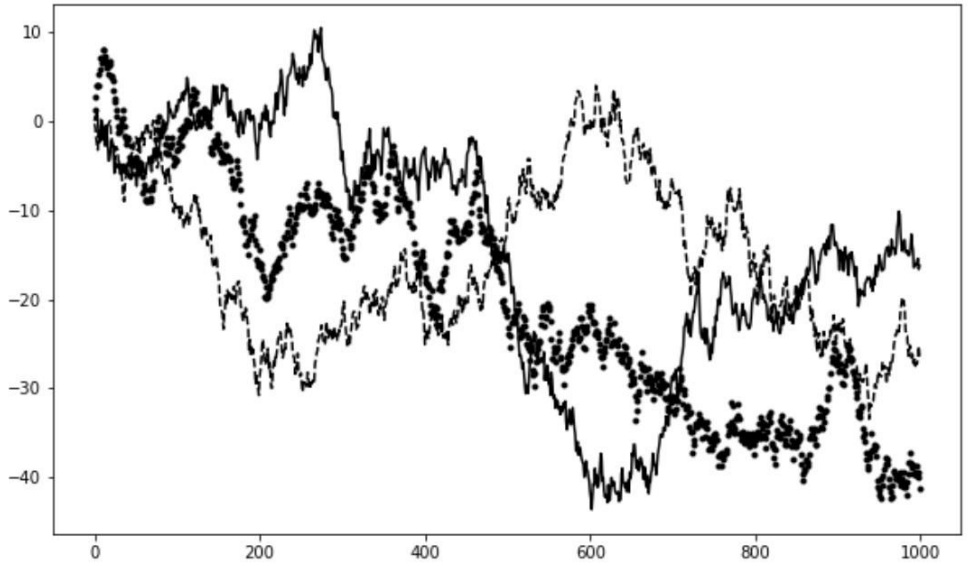
```
Text(0.5, 0, 'Stages')
```



Ex. #12 Adding legends 1

```
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')
```

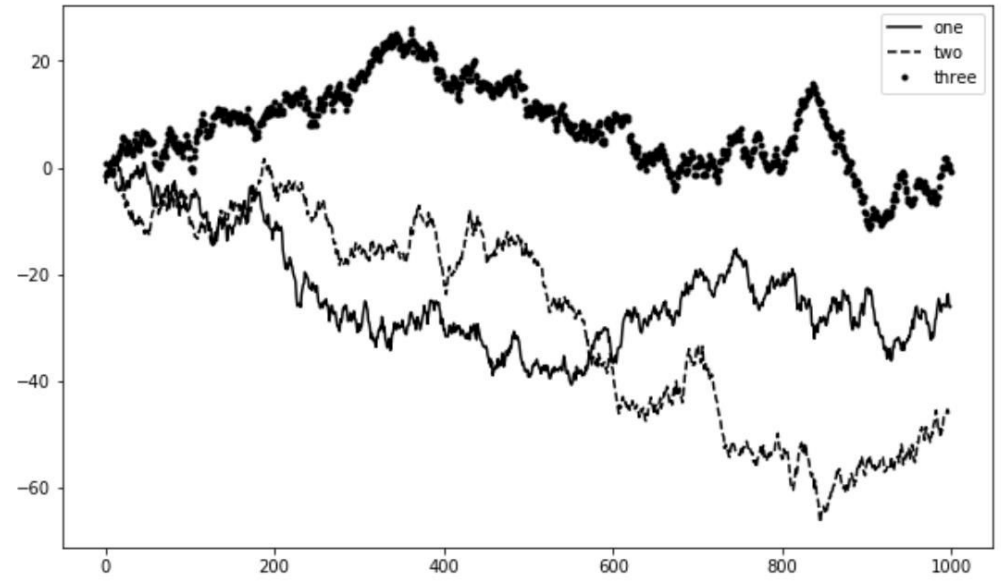
```
[<matplotlib.lines.Line2D at 0x1171597f0>]
```



Ex. #13 Adding legends 2

```
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') # what does this do?
```

```
<matplotlib.legend.Legend at 0x1172b50b8>
```



Are there any other options for the legend? If there are, what are they?

UVA | SCHOOL of DATA SCIENCE

Part IV:
Annotations and Drawing on a Subplot

Ex. #14 Annotations

```
import pandas as pd
from datetime import datetime

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
''' IMPORTANT: Change this path before running this code '''
data =
pd.read_csv('/content/drive/MyDrive/CS5010/code/visualization
/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')
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')
```

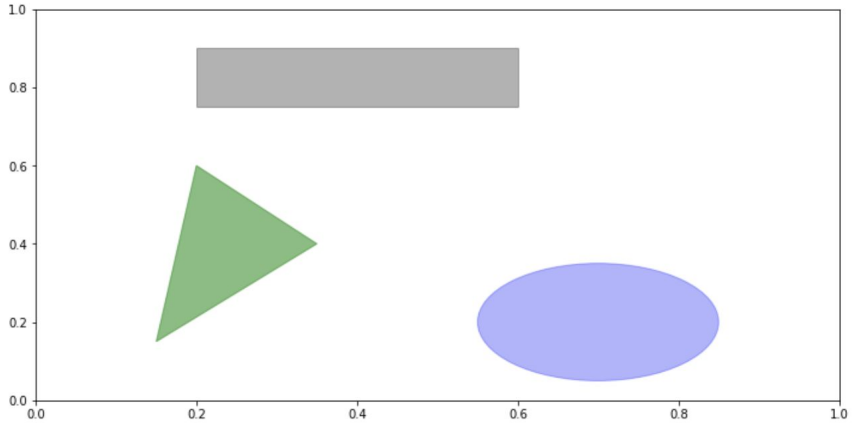
Ex. #14 Annotations

```
Text(0.5, 1.0, 'Important dates in the 2008-2009 financial crisis')
```

Ex. #15 Creating and drawing (plotting) shapes

```
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)
```

```
<matplotlib.patches.Polygon at 0x117f6cc50>
```

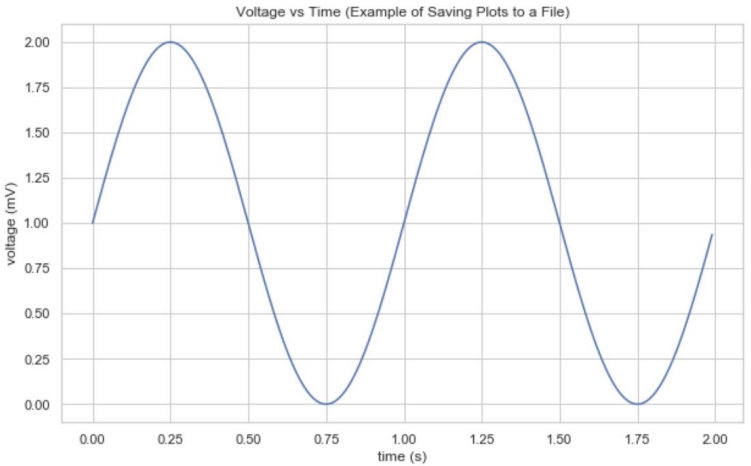


UVA | SCHOOL of DATA SCIENCE

Part V:
Saving Plots to File

Ex. #16 Saving Plots to File

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

t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.plot(t, s)

plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('Voltage vs Time (Example of Saving Plots to a File)')
plt.grid(True)
''' *IMPORTANT: Change this path before running this code!* '''
plt.savefig('/content/drive/MyDrive/CS5010/code/visualization/test.png')
plt.show()
```



UVA | SCHOOL of DATA SCIENCE

Part VI: matplotlib Configuration

matplotlib Configuration

Use of 'rc ' method to modify configuration

- matplotlib comes configured with color schemes and defaults that are geared primarily toward preparing figures for publication.
- Most of these default settings can be customized via an extensive set of global parameters governing figure size, subplot spacing, colors, font sizes, grid styles, and so on.
- One way to modify the configuration is to use the ' rc ' method. For example, to set the global default figure size to be 10x10, enter the following code:

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

- The first argument to rc is the component you wish to customize, such as 'figure', 'axes', ' xtick ', ytick ','grid', 'legend', or many others
- After that can follow a sequence of keyword arguments indicating the new parameters. An easy way to write down the options in your program is as a dict (dictionary)

UVA | SCHOOL of DATA SCIENCE

Part VII:
Plotting with pandas and seaborn

colab

Plotting with pandas and seaborn

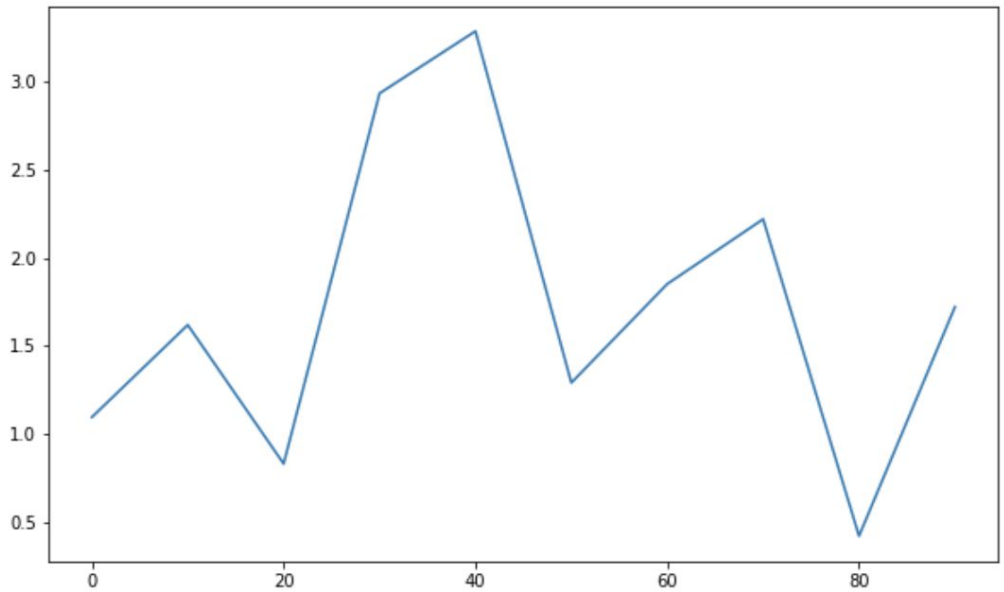
seaborn

- A Python data visualization library based on matplotlib
- provides a high level interface for drawing attractive and informative statistical graphics

Ex. #17 Line Plots 1

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

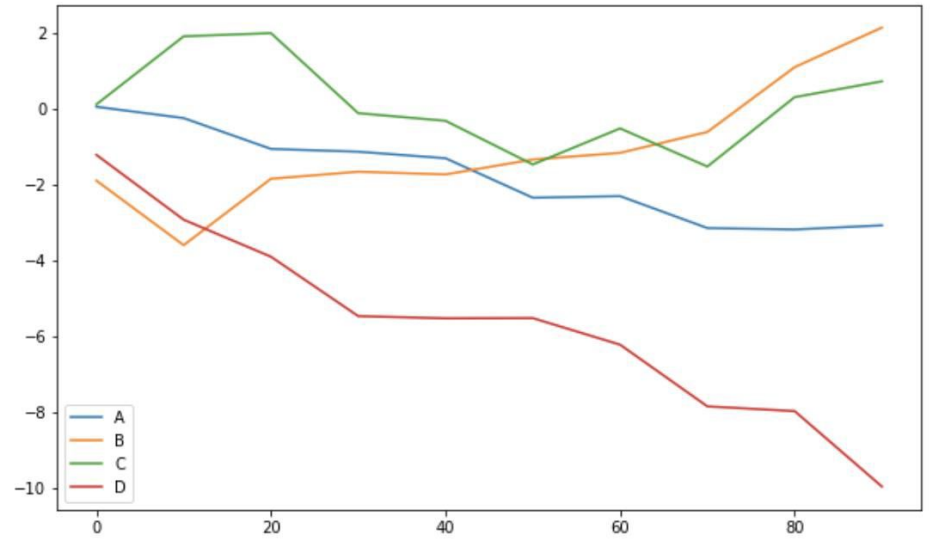
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b13c4a8>
```



Ex. #18 Line Plots 2

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

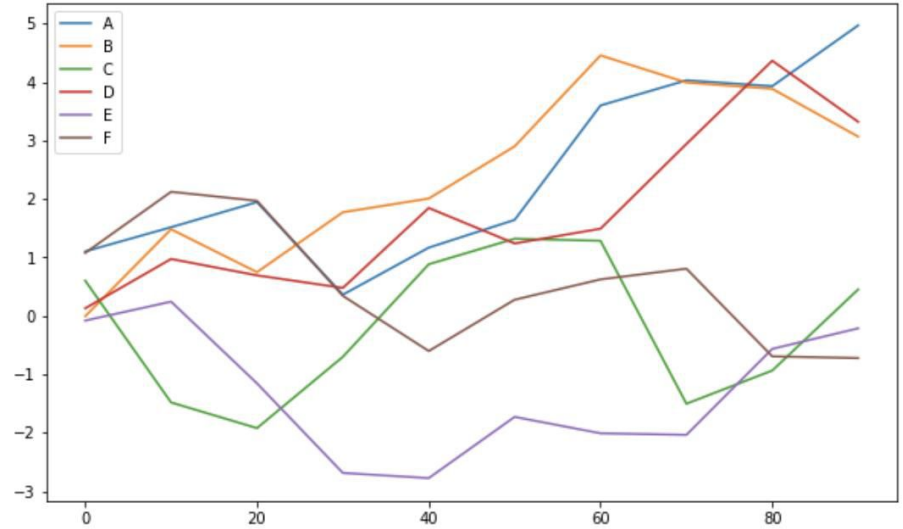
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b28e278>
```



Ex. #19 Line Plots. Another similar example

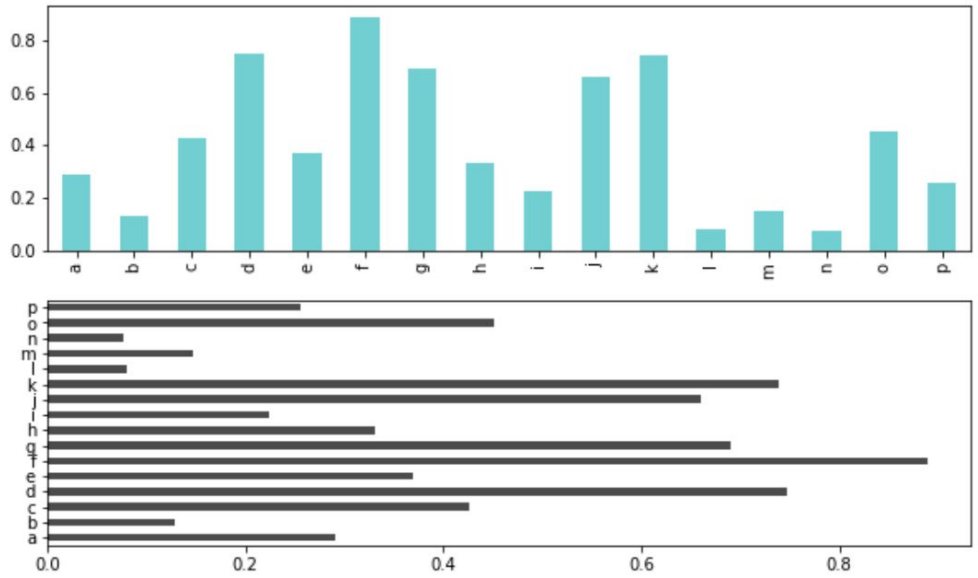
```
df = pd.DataFrame(np.random.randn(10,
6).cumsum(0), # generating 6 plots
                columns=['A', 'B', 'C', 'D', 'E', 'F'],
                index=np.arange(0, 100, 10))
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b42b1d0>
```



Ex. #20 Bar Plots

```
<matplotlib.axes._subplots.AxesSubplot at 0x11ba6b5f8>
```



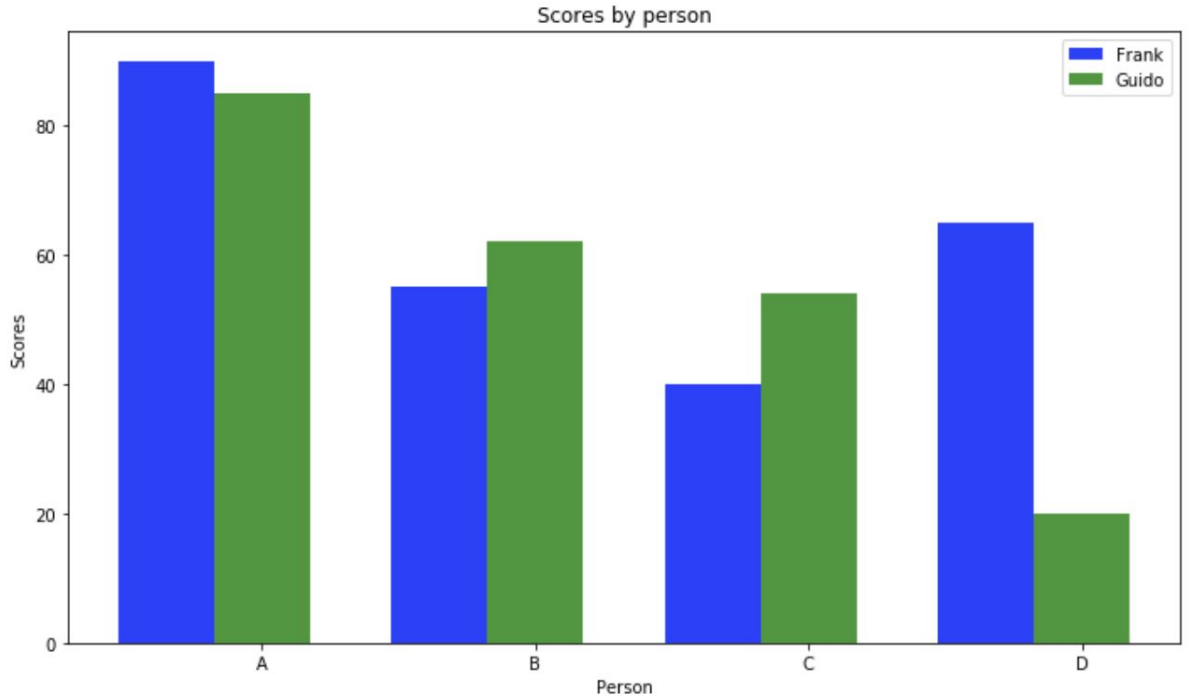

```
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16),
index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='c', alpha=0.7)
data.plot.barh(ax=axes[1], color='k', alpha=0.7)
```

Ex. #21 Bar Plots. Another bar plot example

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

# data to plot
n_groups = 4
means_frank = (90, 55, 40, 65)
means_guido = (85, 62, 54, 20)

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8
rects1 = plt.bar(index, means_frank, bar_width,
alpha=opacity,
color='b',
label='Frank')
```

```
rects2 = plt.bar(index + bar_width, means_guido, bar_width,
alpha=opacity,
color='g',
label='Guido')

plt.xlabel('Person')
plt.ylabel('Scores')
plt.title('Scores by person')
plt.xticks(index + bar_width, ('A', 'B', 'C', 'D'))
plt.legend()

plt.tight_layout()
plt.show()
```

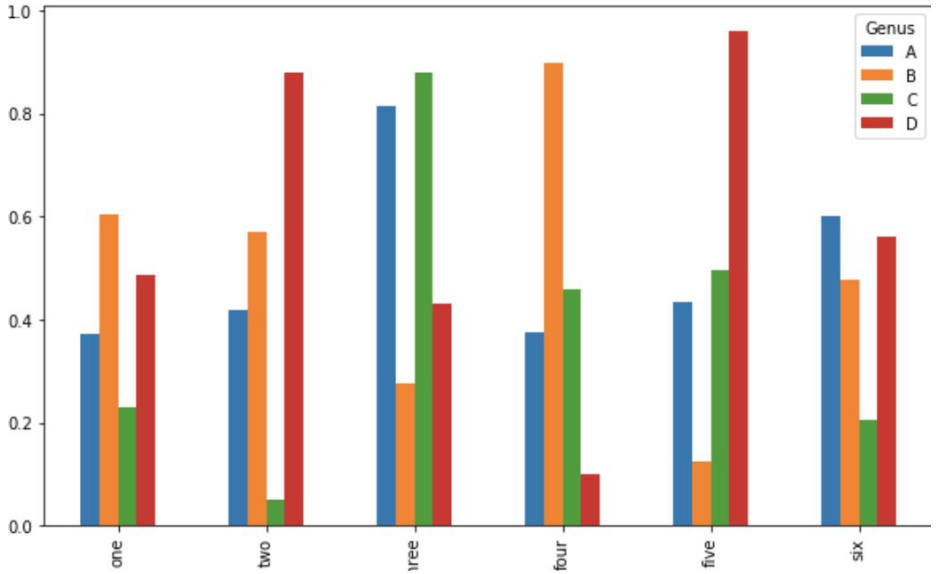
Ex. #21 Bar Plots. Another bar plot example



Ex. #22 Bar Plots. Another bar plot example

```
np.random.seed(12348)

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
df.plot.bar()
```

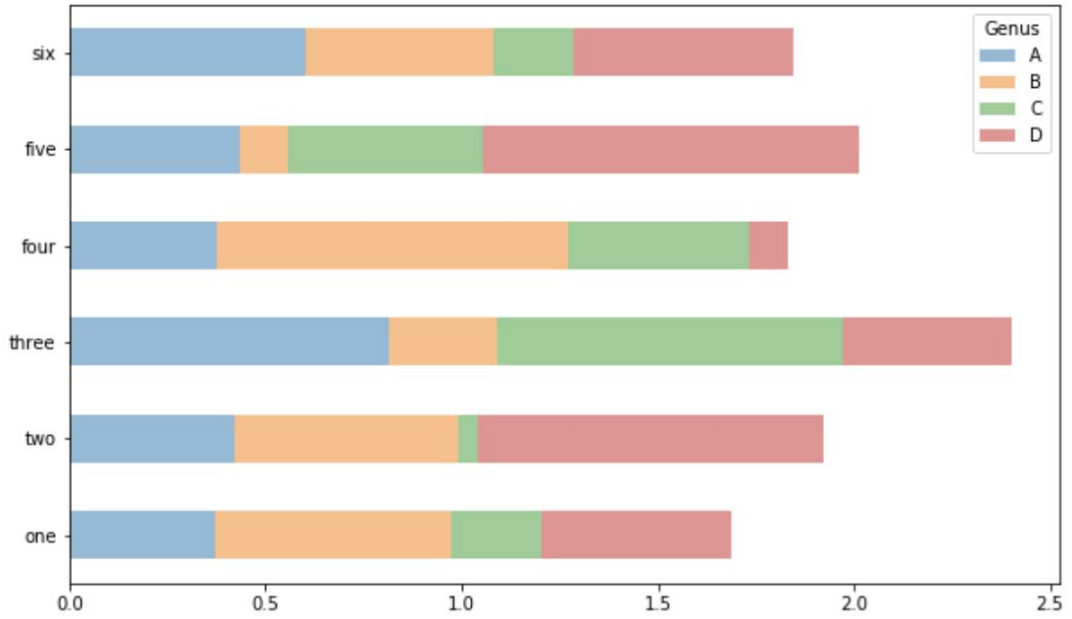
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b0000b8>
```



Ex. #23 Bar Plots. Another bar plot example

```
plt.figure()
df.plot.barh(stacked=True, alpha=0.5)

plt.close('all')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x119b166d8>

<Figure size 720x432 with 0 Axes>
```



Ex. #24 Bar Plots. "tips" data set.

```
tips = pd.read_csv('<Your Path>/tips.csv')
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

pandas.crosstable

- Compute a simple cross tabulation of two (or more) factors.
- By default: computes a frequency table

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

Ex. #24 Bar Plots. "tips" data set.

```
party_counts = party_counts.loc[:, 2:5] # preserve size 2 to 5
party_counts
```

size	2	3	4	5
day				
Fri	16	1	1	0
Sat	53	18	13	1
Sun	39	15	18	3
Thur	48	4	5	1

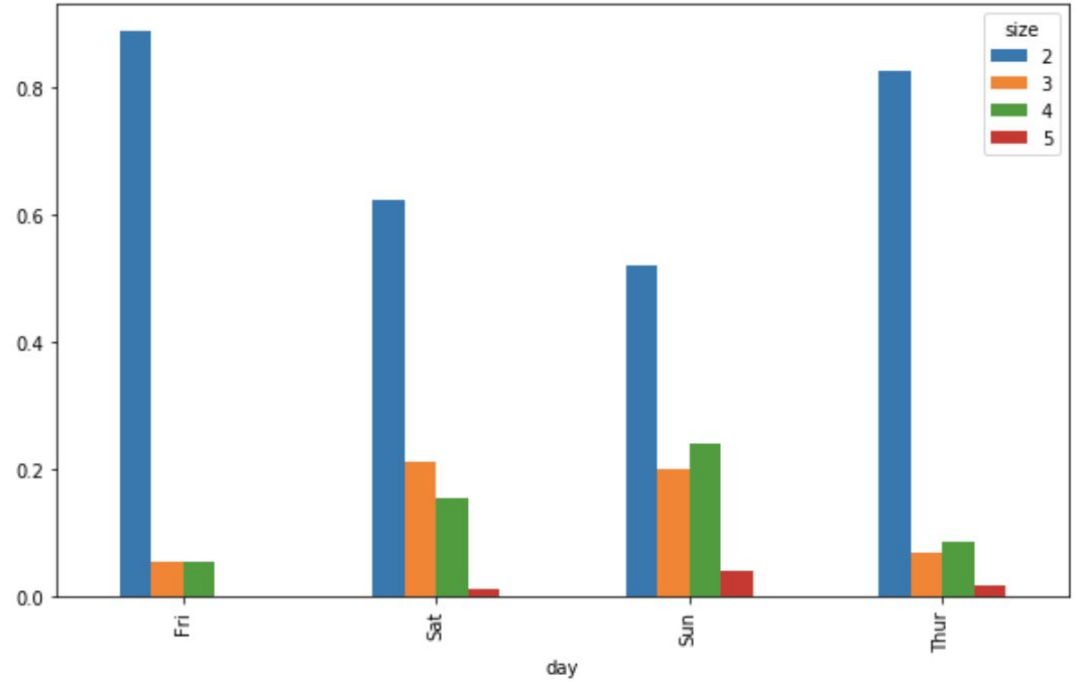
```
# Normalize to sum to 1
party_pcts = party_counts.div(party_counts.sum(1), axis=0)
party_pcts
```

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

Ex. #24 Bar Plots. "tips" data set.

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

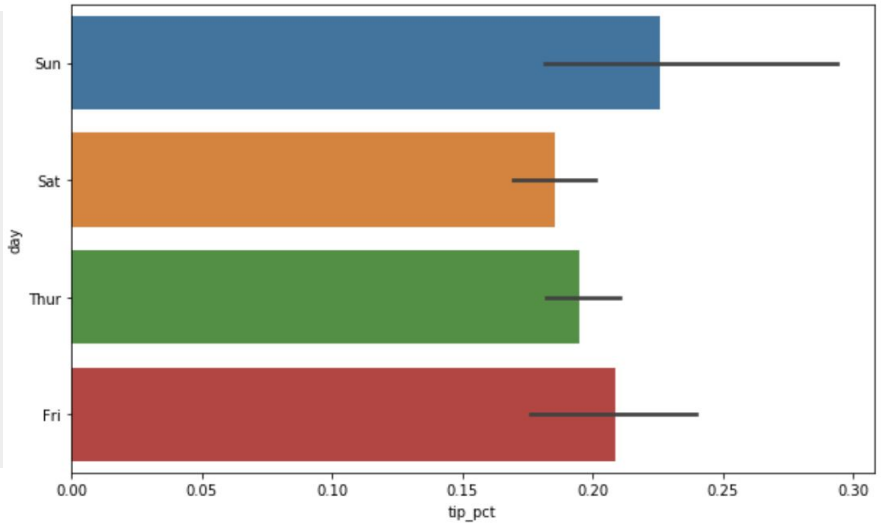
```
<matplotlib.axes._subplots.AxesSubplot at 0x11a88ca58>
```



Ex. #25 Bar plot example using Seaborn 1

```
import seaborn as sns
# tips prior to adding new column
tips.head()
# adding new column to tips
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
tips.head() # short first 5 data items at the top of the file
sns.barplot(x='tip_pct', y='day', data=tips, orient='h')
# Data is tips;  Using tip_pct column
# Organize by 'day'
# Orient the graph horizontally
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d1a9cf8>
```



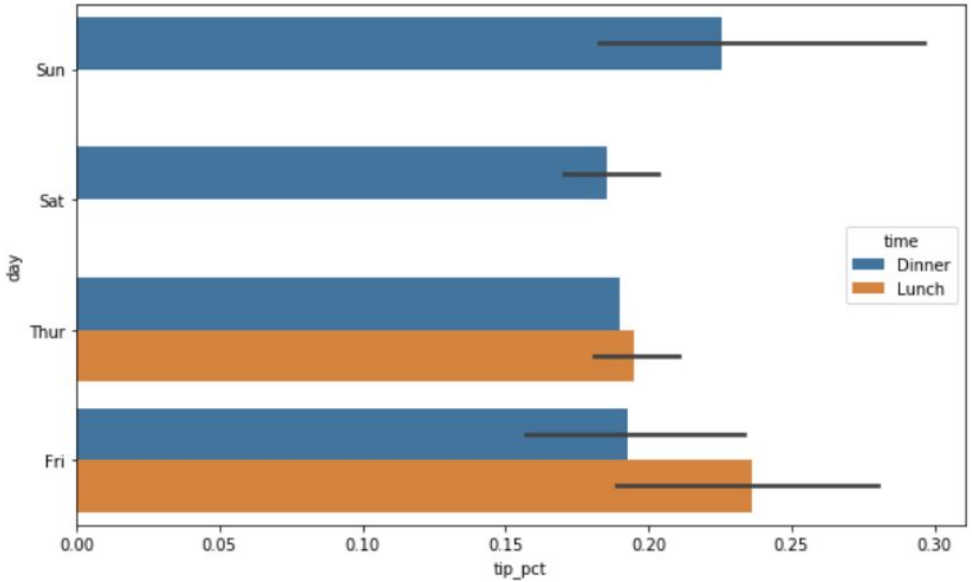
Ex. #26 Plotting with pandas and seaborn

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

Data is tips; using ' tip_pct ' column

Organization: nested grouping by a two variables ('day' and 'time')

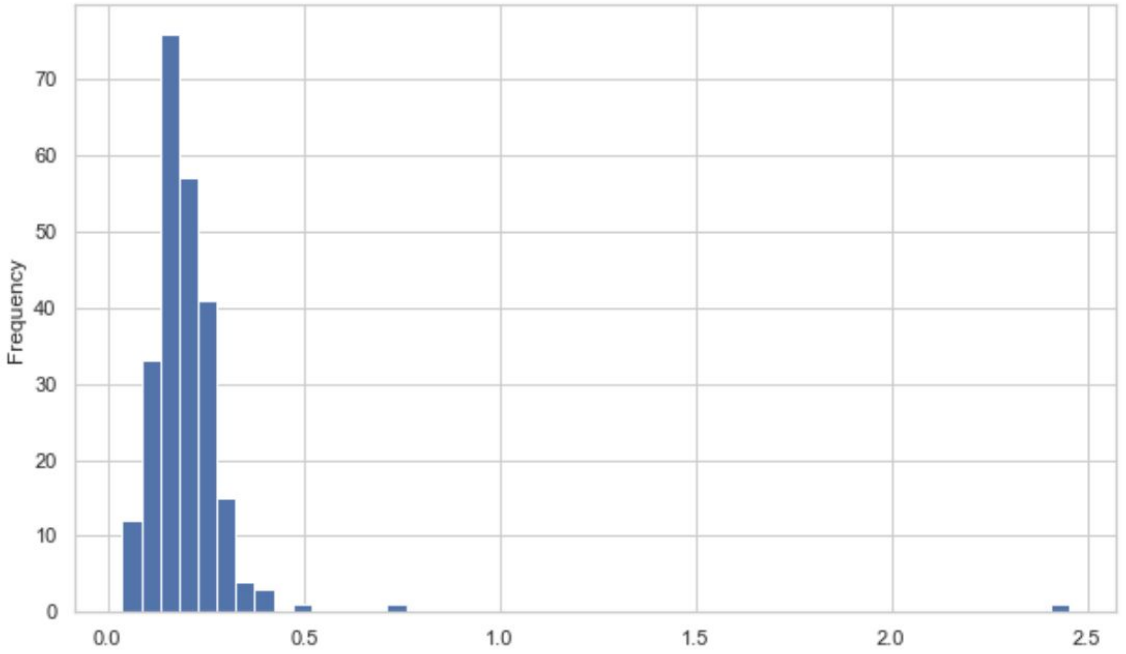
Orient the graph horizontally



Ex. #27 Histogram

```
sns.set(style="whitegrid")
plt.figure()
tips['tip_pct'].plot.hist(bins=50)
```

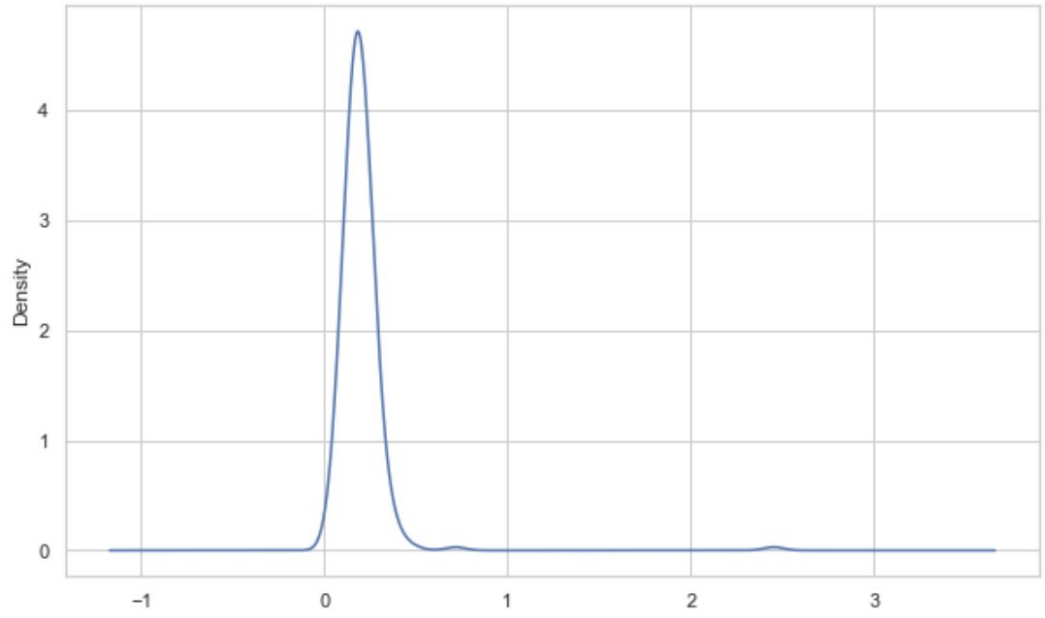
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d3063c8>
```



Ex. #28 Density Plots 1

```
plt.figure()
tips['tip_pct'].plot.density()
```

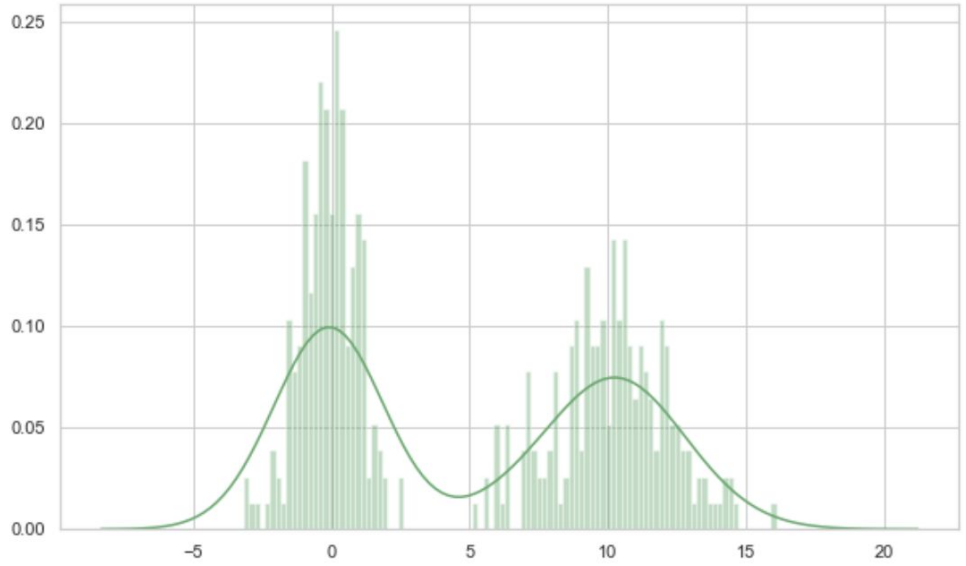
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d4b8550>
```



Ex. #29 Density Plots 2

```
plt.figure()
comp1 = np.random.normal(0, 1, size=200)
comp2 = np.random.normal(10, 2, size=200)
# concatenate the two distributions
values = pd.Series(np.concatenate([comp1,
comp2]))
sns.distplot(values, bins=100, color='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d50cc50>
```



Plotting with pandas and seaborn

Ex. #30-34 Scatter or Point Plots

Next example uses the "macrodata.csv" data set. Ensure the dataset is saved in a location you are aware of!

```
macro = pd.read_csv('<your path>/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
```

	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

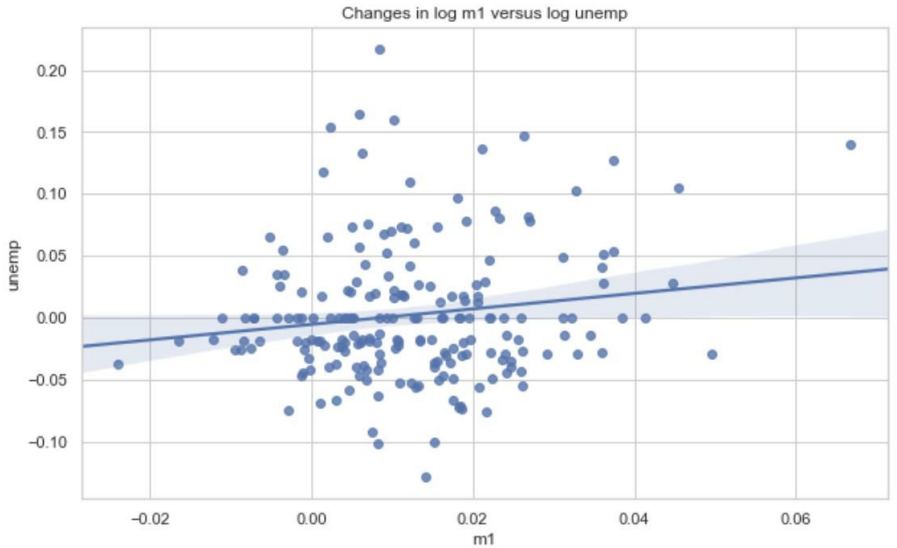
Macroeconomic Data

- cpi: consumer price index
- m1: M1 nominal money stock
- tbilrate average of treasury bill
- umemp unemployment rate (%)

Ex. #30 Scatter Plot

```
plt.figure()
sns.regplot('m1', 'unemp', data=trans_data)  # m1 vs unemp columns to plot
plt.title('Changes in log %s versus log %s' % ('m1', 'unemp'))
```

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

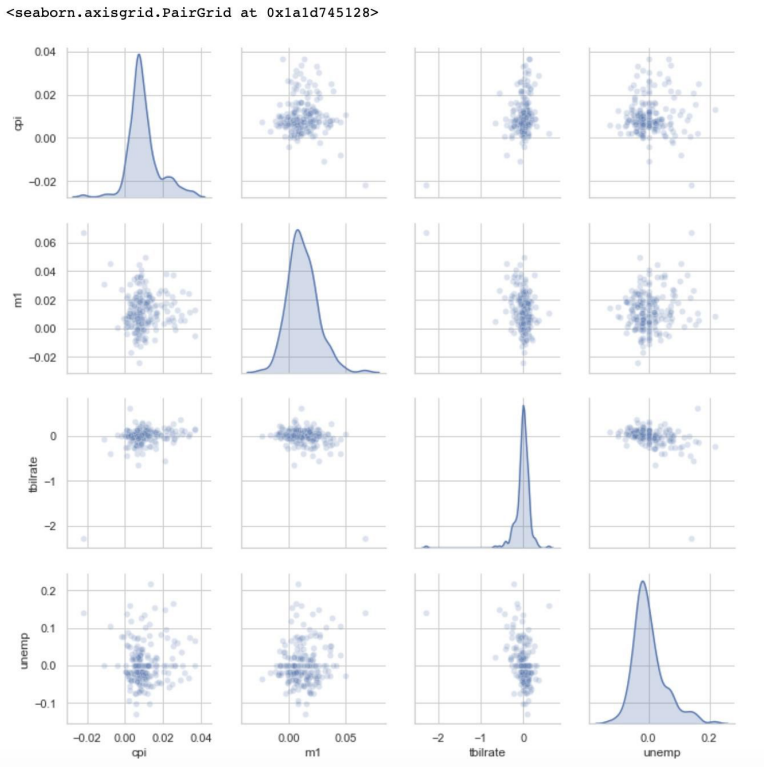


`seaborn.regplot`
Plot data and a linear regression model fit.

Ex. #31 Scatter or Point Plots

Great tool for visualizing the data and comparing attributes (columns) against each other.

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

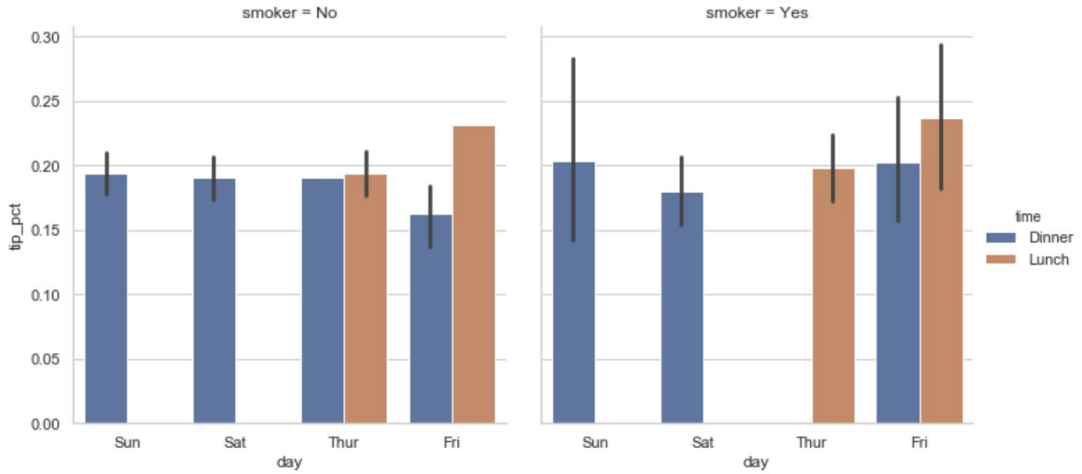


Ex. #32 Facet Grids and Categorical Data 1

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

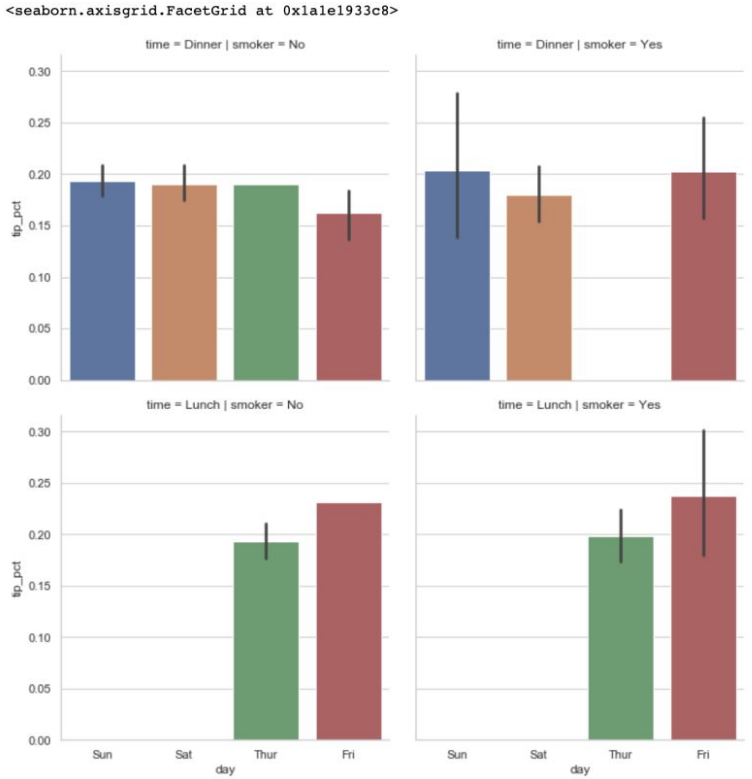
- Using tips data set
- Organization: nested grouping by day and time
- Show results on values of 'smoker' (i.e. no vs yes)
- Using data where tip_pct < 1

```
<seaborn.axisgrid.FacetGrid at 0x1a1d72a518>
```



Ex. #33 Facet Grids and Categorical Data

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



Similar to above, only now organized by combinations of time and smoker : dinner and no, dinner and yes, lunch and no, lunch and yes

Ex. #34 Box plot

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

```
<seaborn.axisgrid.FacetGrid at 0x1a1e764b38>
```