**Useful Codes:**

* plt.tight\_layout(): **this line will magically solve the layout problems for your plot.**
* plt.savefig('plots/xyz.png', dpi=400): **use dpi to generate high resolution plot.**

**Basic Plot Workflow:  
Step1: Prepare Data**   
 Import matplotlib.pyplot as plt  
 x = [1,2,3,4]  
 y=[10,20,25,30]  
  
**Step 2: Create Plot**  
 fig = plt.figure()  
  
**Step3: Plot**  
 ax = fig.add\_subplot(111)  
  
**Step 4: Customize Plot**  
 ax.plot(x,y, color=’lightblue’, linewidth=3)  
 ax.scatter([2,4,6],   
 [5,15,25],  
 color=’darkgreen’,  
 marker=’^’)  
 ax.set\_xlim(1, 6.5)  
  
**Step 5: Save Plot**  
 plt.savefig(‘foo.png’)  
  
**Step 6: Show Plot**  
 plt.show()

**Colors, Color Bars & Color Maps:**plt.plot(x,x,x,x\*\*2,x,x\*\*3)  
ax.plot(x,y, alpha = 0.4)  
ax.plot(x,y, c = ‘k’)  
fig.colorbar(im, orientation=’horizontal’)  
im = ax.imshow(img,  
 cmap=’seismic’)  
  
  
**Markers:**fig, ax = plt.subpots()  
ax.scatter(x,y,marker=”.”)  
ax.plot(x,y,marker=’o’)  
  
**Linestyles:**  
plt.plot(x,y,linewidth=4.0)  
plt.plot(x,y,ls=’solid’)  
plt.plot(x,y,ls=’—’  
plt.plot(x,y,’—‘,x\*\*2,y\*\*2,’-.’)  
plt.setp(lines,color=’r’,linewidth=4.0)  
  
**Test & Annotations:**  
ax.test(1,  
 -2.1,  
 ‘Example Graph’,  
 style=’italic’)  
ax.annotate(“Sine”,  
 xy=(8, 0),  
 xycoords=’data’,  
 xytext=(10.5, 0),  
 textcoords=’data’,  
 arrowprops=dict(arrowstyle=”->”,  
 connectionstyle=”arc3”),)  
  
**Mathtext:**  
plt.title(r’$sigma\_i=15$’, fontsize=20)  
  
**Limits, Legends & Layouts:**  
**Limits & Autoscaling:** ax.margins(x=0.0, y=0.1) Add padding to a plot  
 ax.axis(‘equal’) Set the aspect ratio of the plot to 1  
 ax.set(xlim=[0,10.5], ylim=[-1.5,1.5]) Set limits for x- and y-axis  
 ax.set\_xlim(0,10.5) Set limits for x-axis  
  
**Legends:**  
 ax.set(title=’An Example Axes’, Set a title and x-and y=axis labels  
 ylabel=’Y-Axis’,  
 xlabel=’X-Axis’)  
 ax.legen(loc=’best’) No overlapping plot elements

**Ticks:**  
 ax.xaxis.set(ticks=range(1,5), Manually set x-ticks  
 ticklabels=[3,100,-12,”foo”])  
 ax.ticks\_params(axis=’y’, Make y-ticks longer and go in and out  
 direction=’inout’,  
 ;ength=10)  
  
**Subploting Spacing:** fig3.subplots\_adjust(wspace=0.5, Adjust the spacing between subplots  
 hspace=0.3,  
 left=0.125,  
 right=0.9,  
 top=0.9,  
 bottom=0.1)  
 fig.tight\_layout() Fit subplots() in to the figure area  
 **Axis Spines:**  
 ax1.spines[‘top’].set\_visible(False) Make the top axis line for a plot invisible  
 ax1.spines[‘bottom’].set\_position((‘outward’, 10)) Move the bottom axis line outward  
  
**Plotting Routines:  
1D Data:** lines = ax.plot(x,y) Draw points with lines or markers connecting them  
 ax.scatter(x,y) Draw unconnected points, scaled or colored  
 axes[0,0].bar([1,2,3],[3,4,5]) Plot vertical rectangles (constant width)  
 axes[1,0].barh([0.5,1,2.5],[0,1,2]) Plot horizontal rectangles (constant height)  
 axes[1,1].axhline(0.45) Draw a horizontal line across axes  
 axes[0,1].axvline(0.65) Draw a vertical line across axes  
 ax.fill(x,y,color=’blue’) Draw filled polygons  
 ax.fill\_between(x,y,color=’yellow’) Fill between y-values and o  
  
**2D Data or Images:**  
 fig, ax = plt.subplots() Colormapped or RGB arrays  
 im = ax.imshow(img,  
 cmap=’gist\_earth’,  
 interpolation=’nearest’,  
 vmin=-2,  
 vmax=2)

axes2[0].pcolor(data2) Pseudocolor plot of 2D array  
 axes2[0].pcolormesh(data) Pseudocolor plot of 2D array  
 CS = plt.contour(Y,X,U) Plot contours  
 axes2[2].contourf(data1) Plot filled contours   
 axes2[2] = ax.clabel(CS) Label a contour plot  
  
**Vector Fields:**  
 axes[0,1].arrow(0,0,0.5,0.5) Add an arrow to the axes  
 axes[1,1].quiver(y,z) Plot a 2D field of arrows  
 axes[0.1].streamplot(X,Y,U,V) Plot 2D vector fields  
  
**Data Distributions:** ax1.hist(y) Plot histogram  
 ax3.boxplot(y) Make a box and whisker plot  
 ax3.violinplot(z) Make a violin plot

**Scatter Plot:**

# Pandas for managing datasets  
import pandas as pd  
df = pd.read\_csv('Pokemon.csv', encoding = "ISO-8859-1", index\_col=0)

# Matplotlib for additional customization  
from matplotlib import pyplot as plt  
%matplotlib inline  
import seaborn as sns  
# Recommended way  
sns.lmplot(x='Attack', y='Defense', data=df)

# Alternative way  
# sns.lmplot(x=df.Attack, y=df.Defense)

#Scatterplot arguments  
sns.lmplot(x='Attack', y='Defense', data=df,  
 fit\_reg=False, # No regression line  
 hue='Stage') # Color by evolution stage  
  
# Plot using Seaborn  
sns.lmplot(x='Attack', y='Defense', data=df,  
 fit\_reg=False,   
 hue='Stage')

# Tweak using Matplotlib  
plt.ylim(0, None)  
plt.xlim(0, None)

**Boxplot:**# Boxplot  
sns.boxplot(data=df)

# Pre-format DataFrame  
stats\_df = df.drop(['Total', 'Stage', 'Legendary'], axis=1)

# New boxplot using stats\_df  
sns.boxplot(data=stats\_df)

**Violinplot:**  
# Set theme  
sns.set\_style('whitegrid')

# Violin plot  
sns.violinplot(x='Type 1', y='Attack', data=df)

#Custom color palette  
pkmn\_type\_colors = ['#78C850', # Grass  
 '#F08030', # Fire  
 '#6890F0', # Water  
 '#A8B820', # Bug  
 '#A8A878', # Normal  
 '#A040A0', # Poison  
 '#F8D030', # Electric  
 '#E0C068', # Ground  
 '#EE99AC', # Fairy  
 '#C03028', # Fighting  
 '#F85888', # Psychic  
 '#B8A038', # Rock  
 '#705898', # Ghost  
 '#98D8D8', # Ice  
 '#7038F8', # Dragon  
 ]

#using custom color palette  
# Violin plot with Pokemon color palette  
sns.violinplot(x='Type 1', y='Attack', data=df,   
 palette=pkmn\_type\_colors) # Set color palette

**Heatmap:**

# Calculate correlations  
corr = stats\_df.corr()

# Heatmap  
sns.heatmap(corr)

**Histogram:**

import numpy as np  
%config InlineBackend.figure\_format = 'retina'

np.random.seed(3)  
x = np.random.randn(250)  
plt.hist(x);

# Distribution Plot (a.k.a. Histogram)  
sns.distplot(df.Attack)

**Bar plot:**

# Count Plot (a.k.a. Bar Plot)  
sns.countplot(x='Type 1', data=df, palette=pkmn\_type\_colors)

# Rotate x-labels  
plt.xticks(rotation=-45)

**Joint Distribution Plot:**

# Joint Distribution Plot  
sns.jointplot(x='Attack', y='Defense', data=df)

# First, we'll import pandas, a data processing and CSV file I/O library  
import pandas as pd

# We'll also import seaborn, a Python graphing library  
import warnings # current version of seaborn generates a bunch of warnings that we'll ignore  
warnings.filterwarnings("ignore")  
import seaborn as sns  
import matplotlib.pyplot as plt  
sns.set(style="white", color\_codes=True)

# Next, we'll load the Iris flower dataset, which is in the "../input/" directory  
iris = pd.read\_csv("iris.csv") # the iris dataset is now a Pandas DataFrame

#create id column based on index  
iris['id'] = iris.index  
# Let's see what's in the iris data - Jupyter notebooks print the result of the last thing you do  
iris.head()

**Kdeplot:**

sns.FacetGrid(iris, hue="species", size=6) \  
 .map(sns.kdeplot, "petal\_length") \  
 .add\_legend()

**Pairplot:**

sns.pairplot(iris.drop("id", axis=1), hue="species", size=3)

**INFO 215 Samples:**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import itertools

df = pd.read\_csv('Music Sales.csv', index\_col=None)

df\_EA=df[df['Region'].isin(['Europe', 'Asia'])].copy()  
df\_genre = df\_EA.groupby(['Genre'])['Quantity'].sum().reset\_index()  
df\_genre.sort\_values(by=['Quantity'], ascending=False, inplace=True)

#Revised Chart  
fig, ax = plt.subplots(figsize=(15,10))  
ax = fig.gca()  
ax.xaxis.set\_label\_position('top')  
ax.xaxis.set\_ticks\_position('top')  
sns.despine(bottom=True)  
sns.barplot(y='Genre', x='Quantity', data=df\_genre, color = 'steelblue', ax=ax)  
ax.set\_title('Most Popular Genre in Asia and Europe', size=20)

#Top 20 Artists in Europe and Asia  
# get the rank of artists by quantity  
df\_rank = df\_EA.groupby(['Artist'])['Quantity'].sum().reset\_index()  
df\_rank['Rank'] = df\_rank['Quantity'].rank(method='dense', ascending=False)

# get the quantity by artists and genre  
# append the rank of artists  
df\_rank\_genre = df\_EA.groupby(['Artist', 'Genre'])['Quantity'].sum().reset\_index()  
df\_rank\_genre=pd.merge(df\_rank\_genre, df\_rank[['Artist', 'Rank']], on='Artist', how='left')  
df\_rank\_genre.sort\_values(by='Rank', inplace=True)

#Revised Chart  
df\_top15 = df\_top20.tail(15)  
df\_top15.drop(['Latin', 'Reggae', 'Sci Fi & Fantasy', 'Soundtrack'], axis=1, inplace=True)

ax2 = df\_top15.plot.barh(stacked=True, figsize=(15, 10), cmap=plt.cm.get\_cmap('tab10'))  
for p in ax2.patches:  
 width, height = p.get\_width(), p.get\_height()  
 if width > 0:  
 ax2.annotate(f'{width:.0f}', (p.get\_x()+0.40\*width,  
 p.get\_y()+.17\*height), color='black', fontsize=10)  
plt.title('Top 15 Artists in Asia and Europe', size=20)  
plt.xlabel('Quantity')

#Rock vs Alternative and Punk vs. Heavy Metal  
# format date to month  
df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'])  
df['InvoiceDate\_year\_month'] = pd.to\_datetime(df['InvoiceDate']).dt.strftime("%Y-%m")

# subset the data needed for the plot  
df\_3genres=df[df['Genre'].isin(['Alternative & Punk','Heavy Metal', 'Rock'])]

# get the summary  
df\_3genres\_summary = df\_3genres.groupby(['Genre', 'InvoiceDate\_year\_month'])['Sales'].sum().reset\_index()

# set the template  
t = pd.date\_range('2008-01-01', '2012-12-31', freq='MS')  
order=list(np.arange(60))\*3

df\_3genres\_template=pd.DataFrame(list(itertools.product(['Rock','Alternative & Punk','Heavy Metal'], t)),   
 columns=['Genre', 'InvoiceDate'])  
df\_3genres\_template['order']=order  
df\_3genres\_template['InvoiceDate\_year\_month'] = pd.to\_datetime(df\_3genres\_template['InvoiceDate']).dt.strftime("%Y-%m")

# merge the summary with the template  
df\_3genres\_summary = pd.merge(df\_3genres\_template, df\_3genres\_summary,   
 on=['Genre', 'InvoiceDate\_year\_month'], how='left')

#Revised Chart  
plt.rcParams['figure.figsize'] = (10, 10)  
ax=sns.lmplot(x='order', y='Sales', hue='Genre', data=df\_3genres\_summary, ci=None, markers=['+','o','s'], palette="tab10")  
plt.title('Rock vs. Alternative and Punk vs. Heavy Metal', size=20)  
plt.xlabel('Month')  
ax.set(xticks=np.arange(0,71,12), xticklabels=[2008, 2009, 2010, 2011, 2012, 2013])