Seaborn

```
import seaborn as sns
```

```
#getting seaborn available datasets
sns.get_dataset_names()

data = sns.load_dataset(‘dataset_name’)
```

Seaborn Plots

Categorical	Distribution	Relational	Regression	Matrix
countplot	histplot	scatterplot	regplot	heatmap
barplot	kdeplot	lineplot		
boxplot	rugplot	relplot		
violinplot	ecdfplot			
stripplot	displot			
swarmplot	jointplot			
catplot	pairplot			

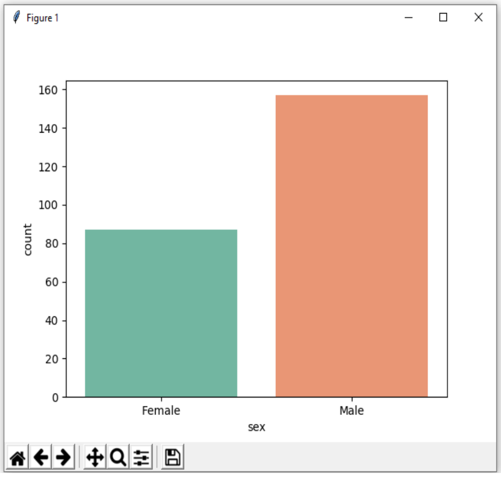
Note:
Hue should be only a categorical variable.
Data specifies dataframe.

x should be usually categoric variable
y should be numeric variable
Changing the order might change the orientation. (*i.e* vertical to horizontal)
x, y need not be provided at the same time. They can be used alone.

Press shift + tab to get function details

Press tab for auto completion

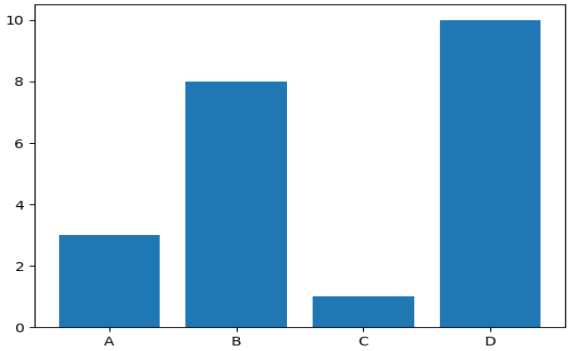
1) Count Plot



```
# vertical plot
sns.countplot(x, data, palette, hue)

# horizontal plot
sns.countplot(y, data)
```

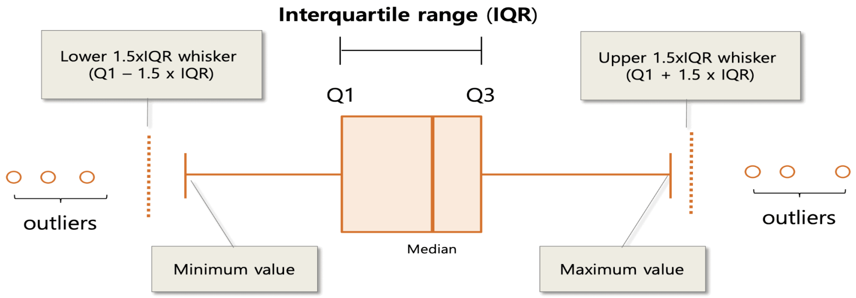
2) Bar Plot



```
sns.barplot(x, y, data, hue, order, hue_order, color, palette, errorbar=None, estimator=len)
```

order - list of columns by which the plot should be plotted.
hue_order - list of values for hue
estimator - np.mean (default), np.median, np.sum, len (actual value)

3) Box Plot

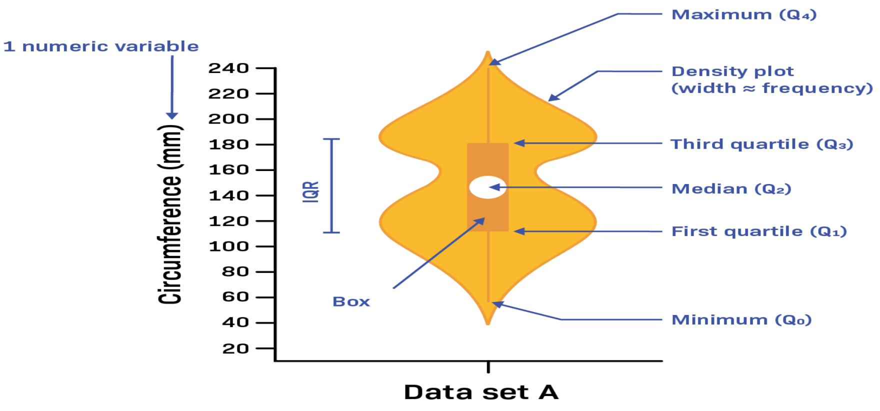


Note:
*Box plot cannot be plotted for categorical variables alone. But it can be plotted for numerical alone
For Categorical - x
For Numerical - y

```
sns.boxplot(x, y, data, linewidth, hue, showmeans=True,
                meanprops={'marker':'o', 'markerfacecolor':'white',
                                 'markeredgecolor':'black', 'markersize':5},
                palette
)

# Box plot for all numerical variables of the dataframe
sns.boxplot(data)
```

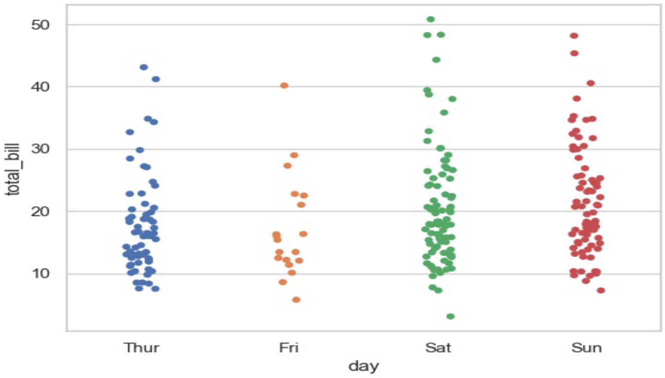
4) Violin Plot



Same note as that of boxplot.

```
sns.violinplot(x, y, data, hue, order, hue_order, color, palette, split=True, bw)
```

5) Strip Plot



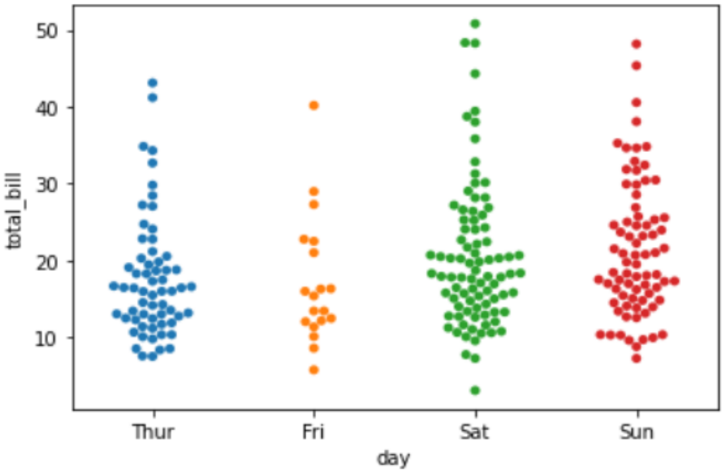
Strip plot can be plotted for both numeric and categorical variables.

sns.stripplot(x, y, data, jitter=0.2, linewidth=0.8, hue, dodge=True, color, palette)

jitter - provides space, dodge=True - separates hue

```
# Drawing strip plot on top of violin plot (also used for box plot)
sns.stripplot(x='Payment', y='Total', data=data, jitter=0.3, palette='Pastel1')
sns.violinplot(x='Payment', y='Total', data=data)
```

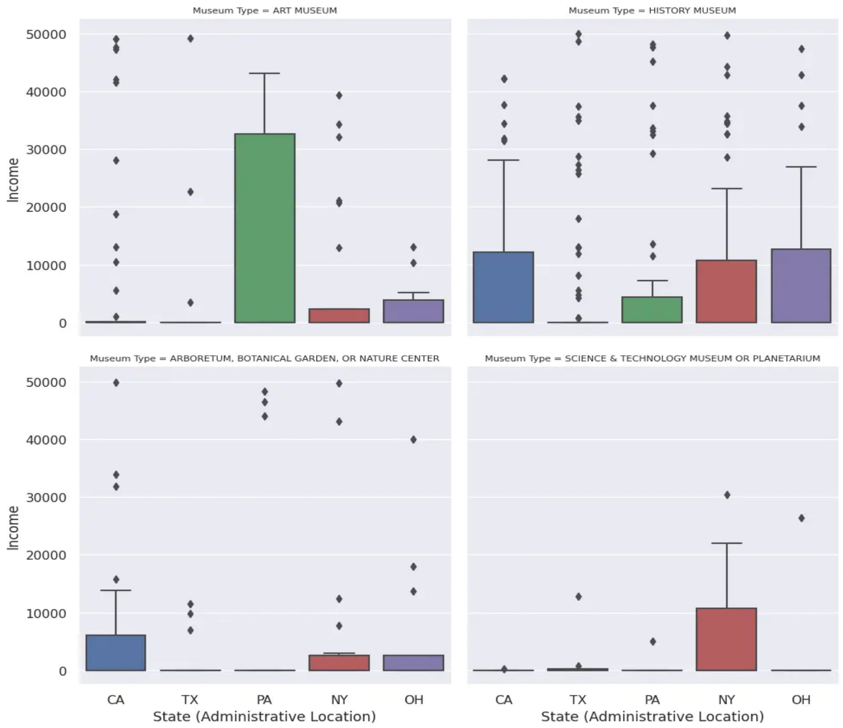
6) Swarm Plot



sns.swarmplot(x, y, data, hue, dodge=True, color, palette, marker, size)

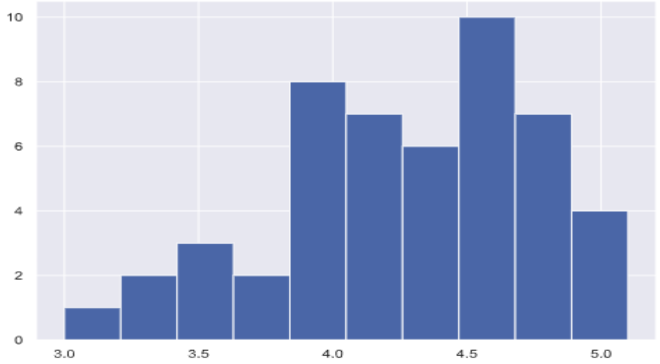
```
# Drawing swarm plot on top of violin plot (also used for box plot)
sns.swarmplot(x='Payment', y='Total', data=data, palette='Pastel1')
sns.violinplot(x='Payment', y='Total', data=data)
```

7) Cat Plot



sns.catplot(x, y, data, kind='bar/violin/box/strip/swarm', hue, row, col, palette)

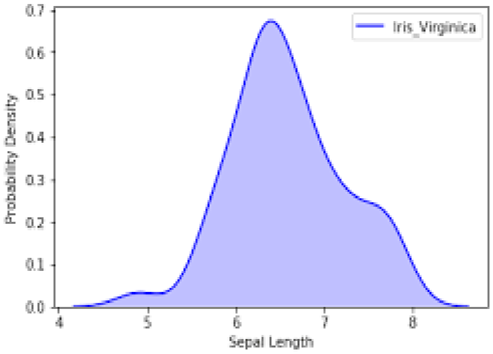
8) Histogram

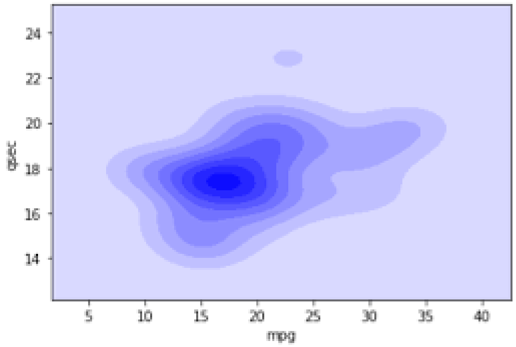


Can be uni as well as bivariate.

```
sns.histplot(x, y, data, binwidth=10, bins=value/list, kde=True, hue, color, palette,
                multiple='stack',
                element='step'/'poly',
                fill=False,
                stat= 'count' (default) /density/ probability/ frequency/ percent )
```

9) KDE Plot (Kernel Density Estimation)





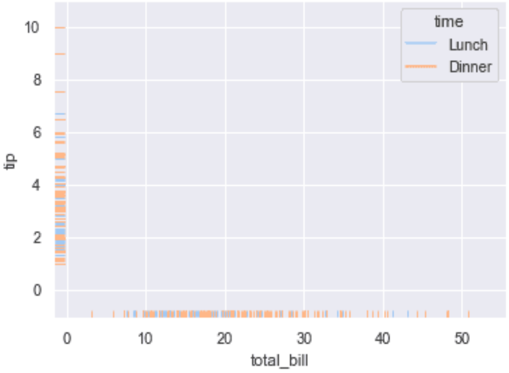
Can be uni as well as bivariate.

sns.kdeplot(x, y, data, fill=True, bw_adjust=0.2, hue, multiple='stack', color, palette, alpha, levels)

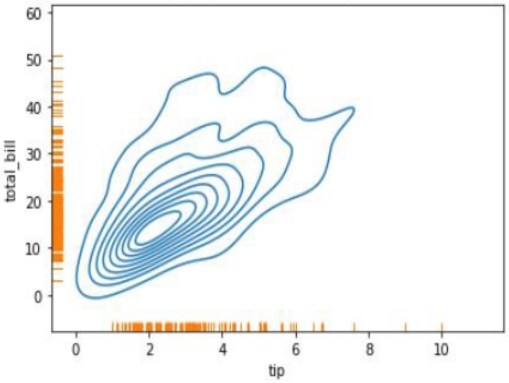
levels - specify the number of contours.

# kde plot for all numeical variables of dataframe
sns.kdeplot(data)

10) Rug Plot



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9d2b4(
```

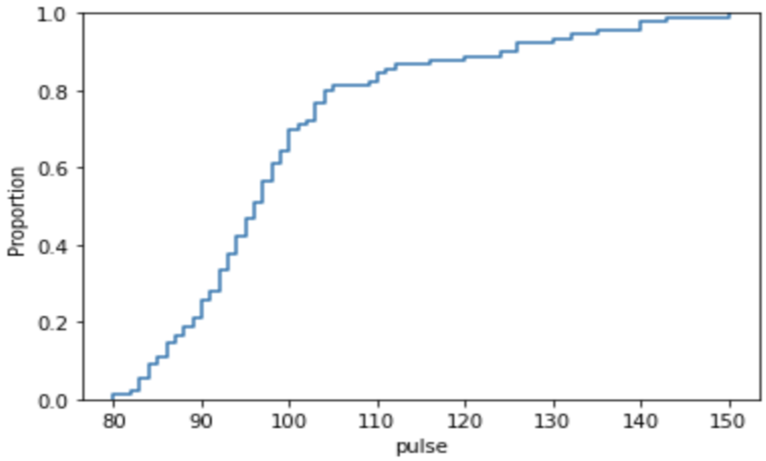


Can be uni/bi variate. Can be used for both categoric and numeric.

sns.rugplot(x, y, data, hue, height=0.1, color)

#Combining with kde plot
sns.rugplot(x='gross income', y='Quantity', data=data, height=0.05)
sns.kdeplot(x='gross income', y='Quantity', data=data, fill=True, color='purple')

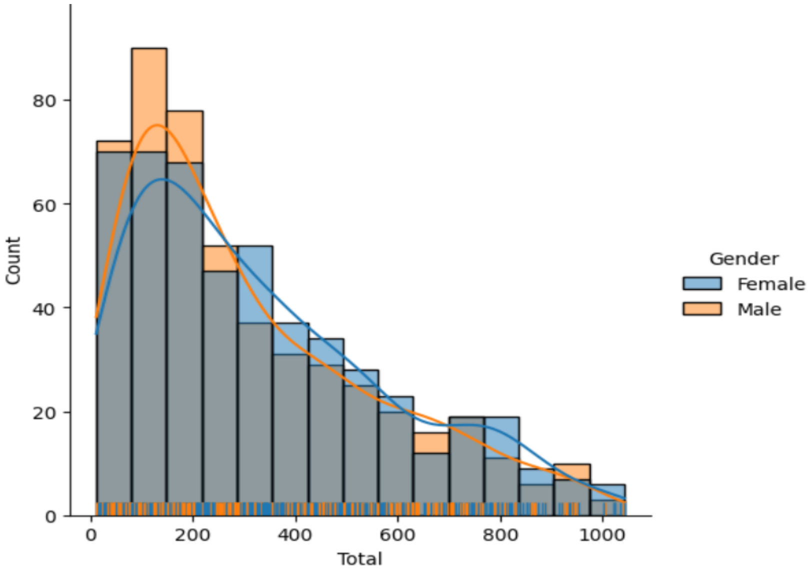
11) ECDF Plot (Empirical Cumulative Distribution Function)



It is a univariate plot. Can be used for both categoric and numeric.

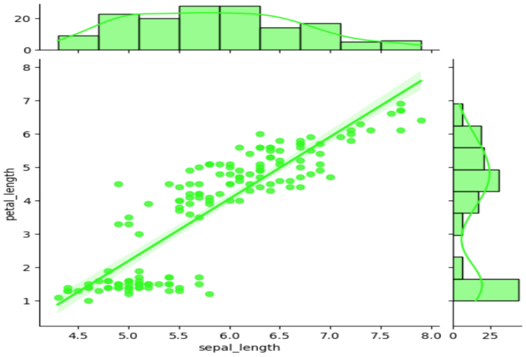
sns.ecdfplot(x, data, hue, color, palette, stat='proportion/count/percent/density')

12) Displot



sns.displot(x, data, kde=True, rug=True, hue, multiple='stack', element='poly', row, col,
color, palette
rug_kws=dict(height=0.1),
kde_kws=dict(bw_adjust=0.12))

13) Joint Plot

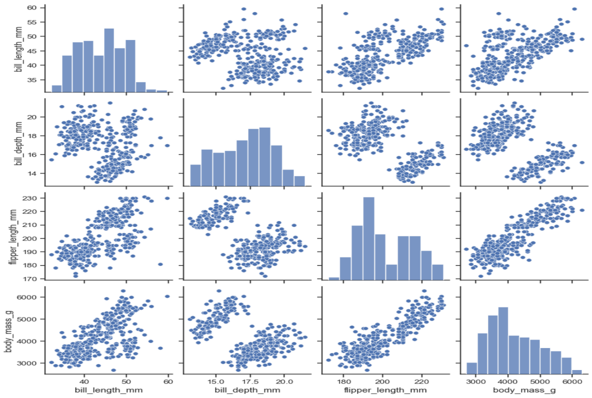


Bivariate plot

```
sns.jointplot(x, y, data, kind=’scatter/hex/kde/resid/hist’, hue, marginal_ticks=True, height=3,
                joint_kws=dict(marker='*', color='red'),
                marginal_kws=dict(color='pink', element='poly'))
```

```
# drawing kde and rugplot on top of joint plot
pl = sns.jointplot(x='petal_length', y='petal_width', data=data, color='#BC2E12',height=4)
pl.plot_joint(sns.kdeplot, color='pink')
pl.plot_joint(sns.rugplot, height=0.1, color='green')
```

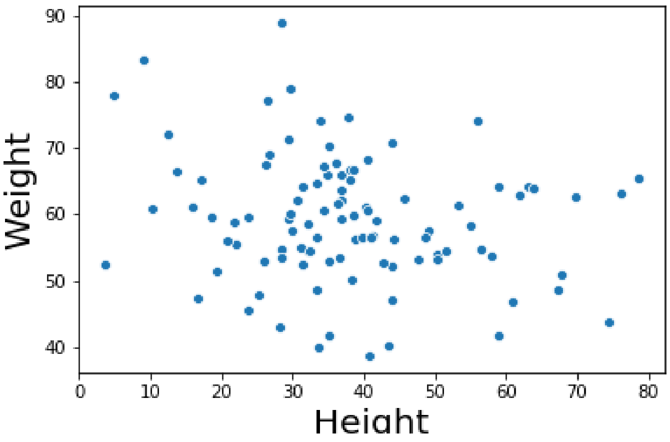
14) Pair Plot



```
sns.pairplot(data, diag_kind=’kde/hist/None’, kind=’scatter/reg/kde/hist’, hue, color, palette,
                x_vars=value/list, y_vars=value/list, corner=True,
                diag_kws=dict(kde=True, color='#16FF00'),
                plot_kws=dict(color='#060047', marker='D', s=5))
```

```
# Creating kde plot on top of pairplot
pl = sns.pairplot(data=data, plot_kws=dict(color='red'), diag_kws=dict(element='poly',
color='pink'))
pl.map_upper(sns.kdeplot)    # For only upper
pl.map_lower(sns.kdeplot, fill=True)   #  For only lower
```

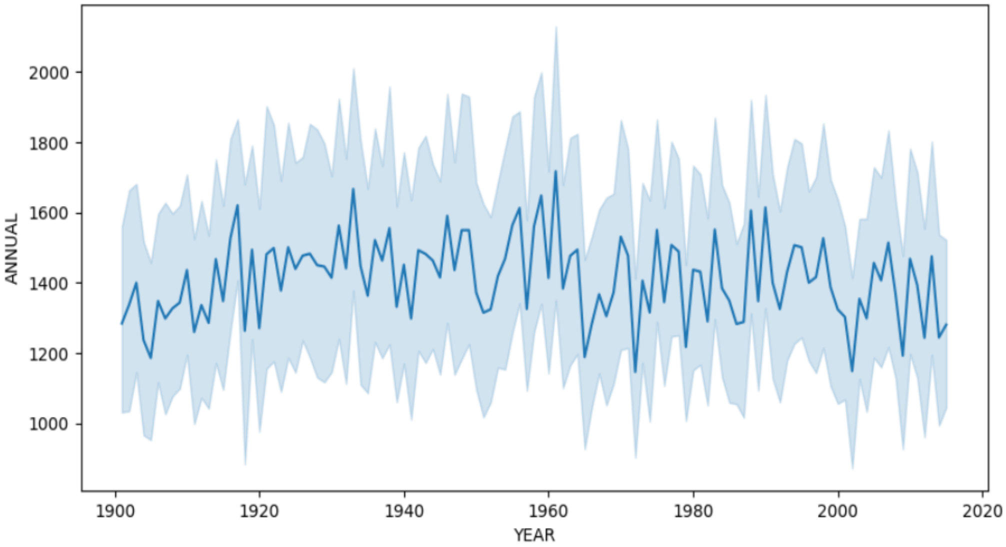
15) Scatter Plot



Bivariate plot

sns.scatterplot(data=data, x, y, hue, palette, s=200, edgecolor=’black’)

16) Line Plot

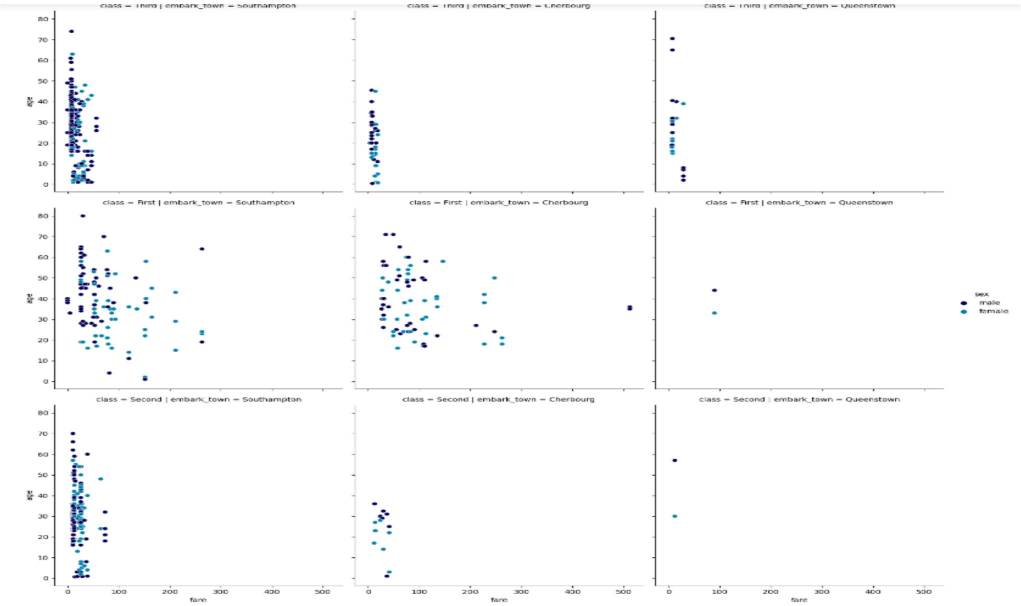


Bivariate Plot

sns.lineplot(data, x, y, errorbar=None, hue, color, palette, estimator=’mean/sum/None’)

None = actual value

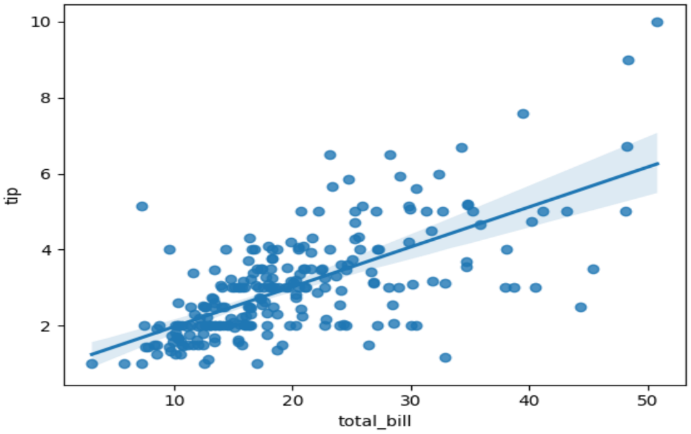
17) Relational Plot



Bivariate, both must be numerical only

sns.relplot(data, x, y, kind=’line/scatter’, errorbar=None, hue, row, col, col_wrap)

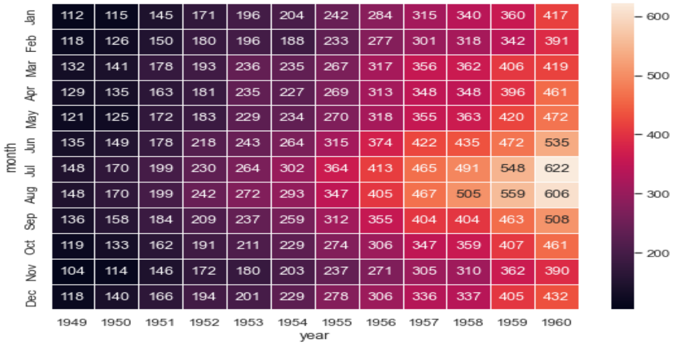
18) Regression Plot



Bivariate Plot.

sns.regplot(data, x, y, color, marker, ci=None,
scatter_kws=dict(color='#5800FF', s=100, alpha=0.5),
line_kws=dict(color='#FF0060', linestyle='--'))

19) HeatMap

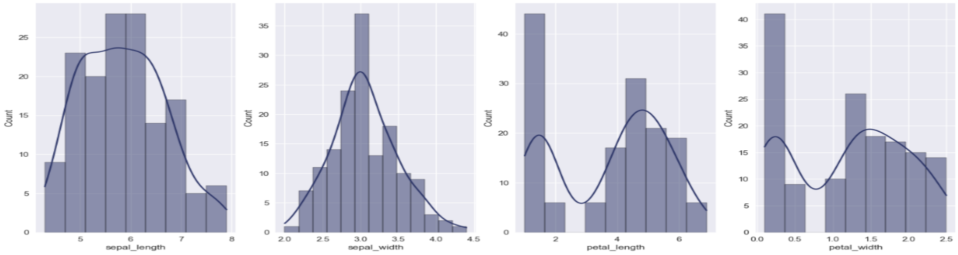


Note:
For heatmap, all columns in the dataframe should be numeric.

```
sns.heatmap(data, annot=True, fmt='.0f', linewidth=0.5, linecolor='white', cmap,
                    annot_kws=dict(size=15, weight='bold'))
```

```
sns.heatmap(data.corr(), vmin=-1, vmax=1, center=0, cmap)
```

20) Subplots



```
# Basic Syntax
fig, ax = plt.subplots(n_rows, n_cols, figsize=(x, y))
```

# 1D 	
fig, ax = plt.subplots(1, 2, figsize=(x, y))
sns.barplot(ax=ax[0], x, y, data)
sns.histplot(ax=ax[1], x, y, data)
	# 1D
	fig, ax = plt.subplots(2, 1, figsize=(x, y))
	sns.barplot(ax=ax[0], x, y, data)
	sns.histplot(ax=ax[1], x, y, data)

```
# 2D
fig, ax = plt.subplots(2, 2, figsize=(x, y))
sns.barplot(ax=ax[0, 0], x, y, data)
sns.histplot(ax=ax[0, 1], x, y, data)
sns.barplot(ax=ax[1, 0], x, y, data)
sns.histplot(ax=ax[1, 1], x, y, data)
```