seaborn

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

Setup

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Importing Data

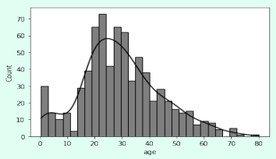
Seaborn offers built in datasets.

```
sns.get_dataset_names()   #returns built in dataset names
titanic_ds = sns.load_dataset('titanic') #loads intended dataset
mpg_ds = sns.load_dataset('mpg')
geyser_ds = sns.load_dataset('geyser')
flights_ds = sns.load_dataset('flights')
```

Plotting

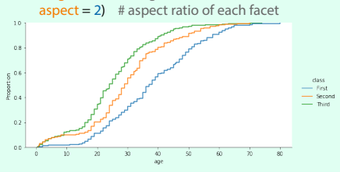
Histogram Plot

```
sns.histplot(                  # creates a histogram plot
      data = titanic_ds , x = 'age' ,   # data set and x axis
      color = 'black',      #color of the bins
      bins = 30,            #number of the bins
      kde = True)           # adds kernel density estimate
```

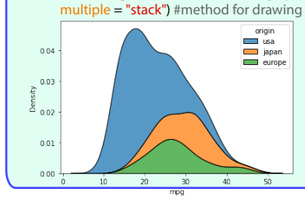


Distribution Plots

```
sns.displot(          #creates a distribution plot in desired kind
                  #ecdf : (empirical cumulative distribution function)
      kind = 'ecdf', #kind of the plot
      data = titanic_ds, x = 'age',  #data set and x axis
      hue = 'class',   # hue is used for color encoding on the plot
      height = 4,    # height of each facet
      aspect = 2)   # aspect ratio of each facet
```

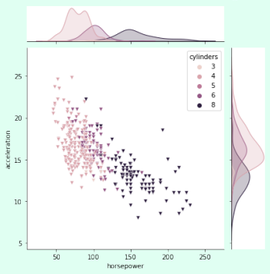


```
sns.kdeplot(                #creates a kernel density estimate plot
      data = mpg_ds,        #data set
      x = "mpg",            #x axis
      hue = "origin",       #color encoding by origin data
      multiple = "stack") #method for drawing multiple elements
```



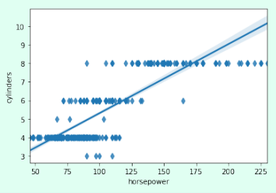
Joint Plot

```
sns.jointplot(
      data = mpg_ds,      #data set
      x = "horsepower",   #x axis
      y = "acceleration",  #y axis
      hue = "cylinders",  #color encoding by cylinders data
      marker = "v")       #determining the marker type
```



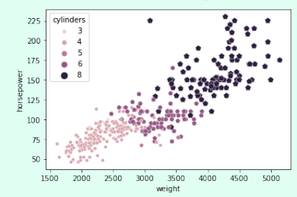
Regression Plot

```
sns.regplot(
      data = mpg_ds,    #data set
      x = "horsepower",  #x axis
      y = "cylinders",    #y axis
      marker = "d")       #determining the marker type
```



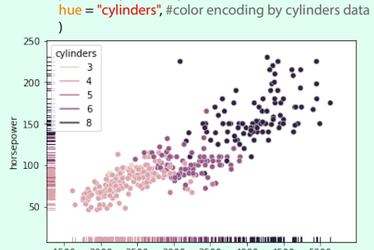
Scatter Plot

```
sns.scatterplot(
      data = mpg_ds,    #data set
      x = "weight",     #x axis
      y = "horsepower",  #y axis
      hue = "cylinders", #color encoding by cylinders data
      size = "cylinders",#determining the size by cylinders data
      marker = "p")     #determining the marker type
                        (p=pentagon)
```



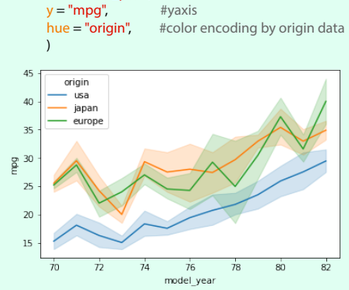
Rug Plot

```
sns.scatterplot(
      data = mpg_ds,    #data set
      x = "weight",     #x axis
      y = "horsepower",  #y axis
      hue = "cylinders", #color encoding by cylinders data
      )

sns.rugplot(
      data = mpg_ds,    #data set
      x = "weight",     #x axis
      y = "horsepower",  #y axis
      hue = "cylinders", #color encoding by cylinders data
      )
```

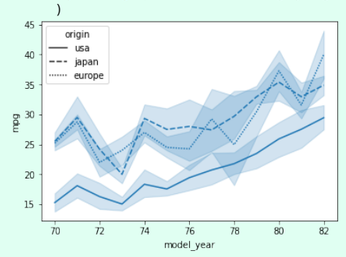


Line Plot

```
sns.lineplot(
      data  = mpg_ds,    #data set
      x = "model_year",  #x axis
      y = "mpg",         #yaxis
      hue = "origin",    #color encoding by origin data
      )
```

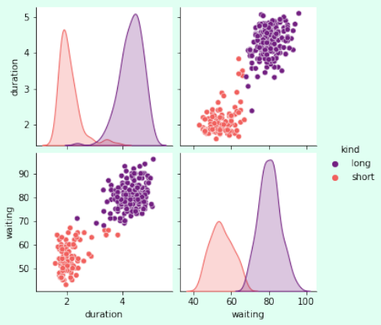


```
sns.lineplot(
      data  = mpg_ds,   #data set
      x = "model_year",  #x axis
      y = "mpg",         #yaxis
      style = "origin",  #pattern encoding by origin data
      )
```



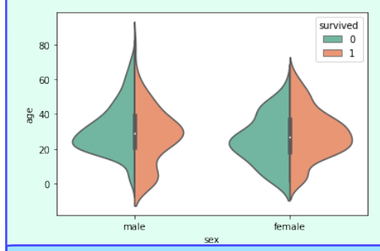
Pair Plot

```
sns.pairplot(
      data = geyser_ds,    #data set
      hue = "kind",        #color encoding by kind data
      palette = "magma")#determining the color palette
```



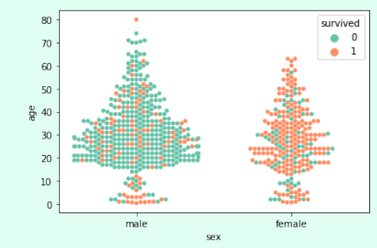
Violin Plot

```
sns.violinplot(
      data = titanic_ds, #data set
      x = "sex",         #x axis
      y = "age",         #y axis
      hue = "survived",  #color encoding by survived data
      split = True,      #when true, draws half of the violin for each
                         #level
      palette = "Set2"   #determining color palette
      )
```



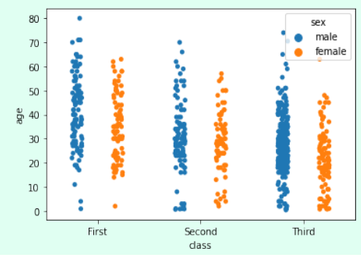
Swarm Plot

```
sns.swarmplot(
      data = titanic_ds, #data set
      x = "sex",         #x axis
      y = "age",         #y axis
      hue = "survived", #color encoding by survived data
      palette = "Set2"   #determining color palette
      size = 4.45        #determining the size of markers
      )
```



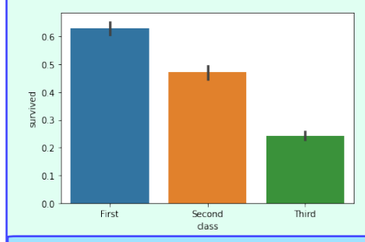
Strip Plot

```
sns.stripplot(
      data = titanic_ds, #data set
      x = "sex",         #x axis
      y = "age",         #y axis
      hue = "sex",  #color encoding by sex data
      dodge = True,      #when true, it seperates the strips
      )
```



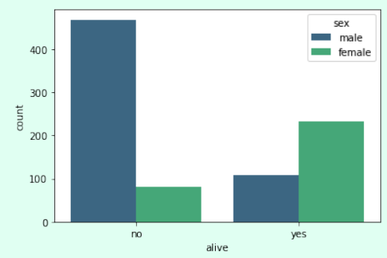
Bar Plot

```
sns.barplot(
      data = titanic_ds, #data set
      x = "class",       #x axis
      y = "survived",    #y axis
      ci = 50            #confidence interval
      )
```



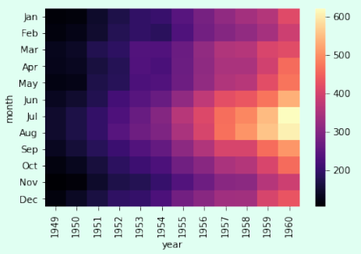
Count Plot

```
sns.countplot(
      data = titanic_ds, #data set
      x = "alive",       #x axis
      hue = "sex", #color encoding by survived data
      palette = "viridis"  #determining color palette
      )
```



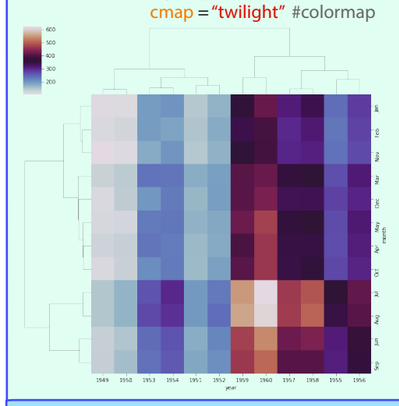
Heat Map

```
flights_ds = flights_ds.pivot("month", "year", "passengers")
#with using dataset.pivot() function dataset is reshaped by given
order
sns.heatmap(
      data = flights_ds,  #dataset
      cmap = "magma")   #determining the colormap
```



Cluster Map

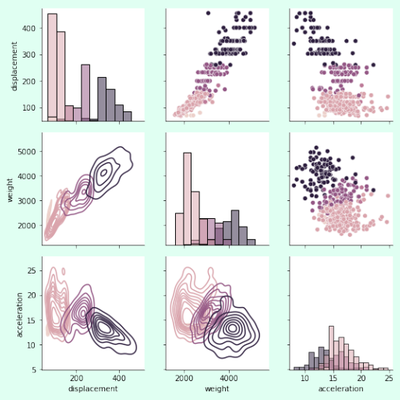
```
sns.clustermap( flights_ds,     #dataset
                cmap = "twilight" #colormap
```



Pair Grid

Pair grid works similar to pair plot, but more flexible.

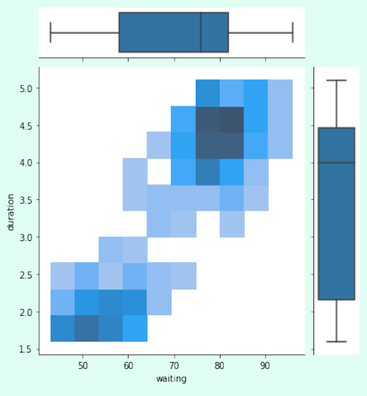
```
grid = sns.PairGrid(mpg_ds, #dataset
      hue = "cylinders")  #encoding colors by cylinders data

grid.map_upper(sns.scatterplot)
#mapping upper part of the grid with scatter plot

grid.map_lower(sns.kdeplot)
 #mapping lower part of the grid with kde plot

grid.map_diag(sns.histplot)
#mapping diagonal part of the grid with histogram
```



Joint Grid

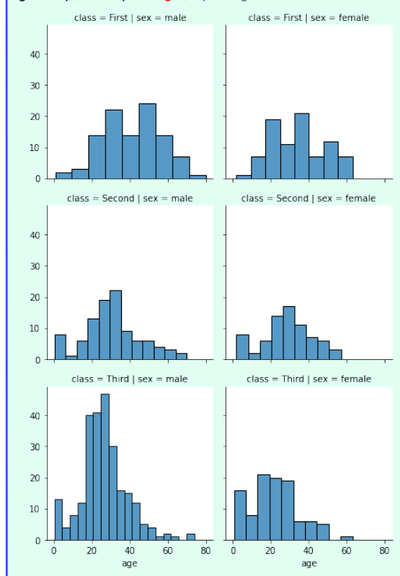
Joint grid works similar to joint plot, but more flexible.

```
grid = sns.JointGrid(
                  data = geyser_ds,   #data set
                  x = "waiting", y = "duration")  , #x and y axis
grid.plot(sns.histplot, sns.boxplot)
#determining which plots to use
```



Faced Grid

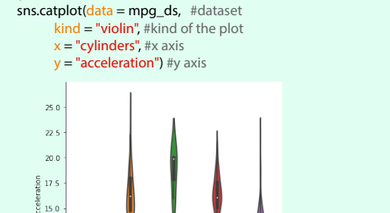
```
grid = sns.FacetGrid(titanic_ds, #dataset
            col = "sex", row = "class")#determining coulmn and rows
grid.map(sns.histplot, "age")  #passing a function and data
```



Relational Plot

relplot() is used for drawing relational plots onto a FacetGrid.

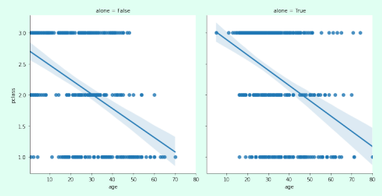
```
sns.relplot(data = titanic_ds,      #dataset
            col = "sex",    #determining row
            x = "class", y= "age") #x and y axis
```



Implot

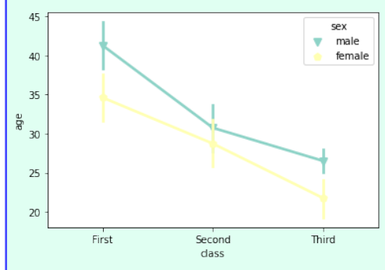
Implot() is combination of regplot() and FacetGrid.

```
sns.lmplot(data = titanic_ds,      #dataset
            col = "alone",    #determining row
            x = "age", y= "pclass") #x and y axis
```



Point Plot

```
sns.pointplot(data = titanic_ds,   #dataset
            x = "class", y = "age", #x, y axis
            hue = "sex",          #color encoding
            markers = ["v", "p"],   #determining markers
            palette = "Set3")     #determining color palette
```



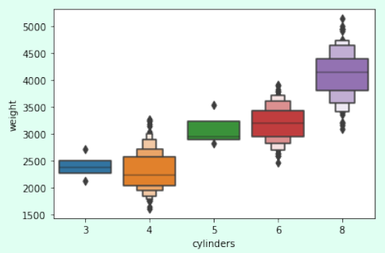
catplot

catplot() is used for creating categorical plots.
It can create box, boxen, strip, swarm, violin, point, bar, count plots.

```
sns.catplot(data = mpg_ds,   #dataset
            kind = "violin", #kind of the plot
            x = "cylinders", #x axis
            y = "acceleration") #y axis
```


Boxen Plot

```
sns.boxenplot(data = mpg_ds,   #dataset
      x = "cylinders", #x axis
      y = "weight") #y axis
```



Styling

sns.set_style

```
# used for changing style
sns.set_style("white")
sns.set_style("dark")
sns.set_style("whitegrid")
sns.set_style("ticks")

sns.set_style("darkdgrid",
  {"grid.color": ".4", #for changing grid color
   "grid.linestyle": "--" #for changing grid line style
        })
```

sns.set_context

```
# used for controlling the scale of the plot elements
sns.set_context("paper")
sns.set_context("poster")

sns.set_style("notebook",
      font_scale = 2.5,     #for controlling the font size
    rc={"lines.linewidth": 3}) # for controlling line witdth
```

sns.despine

```
#used for removing  spines from plot(s)
sns.despine(left = True, bottom = True)
#removes left and bottom spine from plot
sns.despine(right= True, top= True)
```

sns.color_palette

```
#returns desired color palette
sns.color_palette("pastel")

#used for defining a color palette
sns.set_palette("pastel", 3)
```

Setting Axis Labels and Limits

```
plot = sns.lineplot(
      data  = mpg_ds,   #data set
      x = "model_year",  #x axis
      y = "mpg",         #yaxis
      hue = "origin",    #color encoding by
origin data
      )
plot.set_xlabel("Model year" , #changing the label
            fontsize = 20) #changing the font size
plot.set_ylabel("Miles Per Gallon" ,
            fontsize = 20)

plot.xlim(0,150) #setting x axis limit
```

Displaying, Saving and Clearing Plot

```
plot.show() #displays the plot
plot.savefig("lineplot.png", #saves the plot
            transparent = True #saves as transparent
plot.cla() #clears current axis
plot.clf() #clears the figure
```

Closing the Plot

```
plot.close() #closes the plot
```


Emircan Furkan Bayendur

Manisa Celal Bayar University
Computer Engineering Department