SEABORN TUTORIAL

Seaborn is a library for making statistical graphics in Python.

loading dataset

importing libraries

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
```

In [ ]:
```
# current version generates warnings that we can ignore
warnings.filterwarnings("ignore")
```

loading built in dataset

Iris dataset has 50 samples from each of three species of Iris flower (Setosa, Virginica and Versicolor). Four features were measured (in centimetres) from each sample: Length and Width of the Sepals and Petals`

In [ ]:
```
iris=sns.load_dataset('iris')
```

In [ ]:
```
iris.head()
```

In [ ]:
```
iris.shape
```

In [ ]:
```
iris.describe()
```

creating a swarm plot

A swarm plot is a type of scatter plot that is used for representing categorical values

In [ ]:
```
sns.set() #loads default settings of seaborn plot
%matplotlib inline
```

In [ ]:
```
sns.swarmplot(x="species", y="petal_length", data=iris)
```

controlling aesthetics

- One of the biggest advantages of Seaborn over Matplotlib is that its default aesthetics are visually far more appealing.
- Seaborn splits Matplotlib parameters into two independent groups:
  - Styling- sets the aesthetic style of the plot
  - Scaling- scales various elements of the figure to get easily incorporated into different contexts

In [ ]:
```
tips = sns.load_dataset("tips")
tips.head(3)
```

In [ ]:
```
sns.set()
sns.barplot(x = "day", y = "total_bill", data = tips)
```

Styling

In [ ]:
```
sns.set_style("whitegrid")
sns.boxplot(x = "day", y = "total_bill", data = tips)
```

In [ ]:
```
# another style
sns.set_style("ticks")
sns.boxplot(x="day", y="total_bill", data=tips)
```

removing top and right axis

In [ ]:
```
sns.boxplot(x="day", y="total_bill", data=tips)
sns.despine()
# to remove left axis sns.despine(left=True)
```

visualise 2 types of background in same plot

In [ ]:
```
# This function will help us plot some offset sine waves:
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 7):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

with sns.axes_style("darkgrid"):
    plt.subplot(211)
    sinplot()
plt.subplot(212) # this is ticks style
sinplot(-1)
```

Scaling

In [ ]:
```
sns.set()
```

In [ ]:
```
sns.set_context("paper")
sns.set_style("whitegrid")
sns.boxplot(x = "day", y = "total_bill", data = tips)
```

Jupyter Notebook scales down large images in the notebook cell output. This is generally done because past a certain size, we get automatic figure scaling.

In [ ]:
```
sns.set(style='whitegrid', rc={"grid.linewidth": 1.5}) # rc is used for finer
sns.set_context("poster", font_scale=0.5, rc={"lines.linewidth": 3.0})
sinplot()
```

histogram

In [ ]:
```
sns.set()
x = np.random.normal(size=100)
sns.histplot(x,bins=20,kde=True,edgecolor='k')
```

In [ ]:
```
sns.distplot(x, bins=20, kde=True, rug=True, hist_kws=dict(edgecolor='k', line
```

changing axis limit

In [ ]:
```
# Loading dataset
titanic=sns.load_dataset('titanic')

# creating distribution plot
sns.set_style('whitegrid')
sns.histplot(titanic.age.dropna(),kde=True,edgecolor='k')
```

to plot graph with x axis starting at 0

In [ ]:
```
plt.xlim([0,100])
sns.histplot(titanic.age.dropna(),kde=True,edgecolor='k')
```

linear regression plot

- linear regression computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

In [ ]:
```
# loading dataset
tips=sns.load_dataset('tips')
sns.regplot(x='total_bill',y='tip',data=tips,color='b')
```

Pair plot

- A pair plot is a grid of scatter plots where each variable is plotted against every other variable. The diagonal of this grid contains histograms of the individual variables.

In [ ]:
```
data=np.random.random((100,5)) # array with 100 rows and 5 columns with values
columns=['Variable 1', 'Variable 2', 'Variable 3', 'Variable 4', 'Variable 5']
df=pd.DataFrame(data,columns=columns)

# making pair plot
sns.set_style('ticks')
sns.pairplot(df)
```

In [ ]:
```
# customizing fraphs to be displayed
ax=sns.pairplot(df)
ax.map_upper(plt.scatter)
ax.map_lower(sns.kdeplot)
ax.map_diag(sns.distplot)
```

facet grid

In [ ]:
```
bins=np.arange(0,65,5)
x = sns.FacetGrid(tips, col="time",  row="smoker")
x =x.map(plt.hist, "total_bill", bins=bins, color="g")
```

In [ ]:

Color Palettes

In [ ]:
```
sns.set(rc={"figure.figsize": (2, 2)})
current_palette=sns.color_palette()
sns.palplot(current_palette)
```

some diffrent color palettes

In [ ]:
```
sns.palplot(sns.color_palette('hls',10))
```

In [ ]:
```
sns.palplot(sns.color_palette('husl',10))
```

cubehelix color palette

In [ ]:
```
# default palette
sns.palplot(sns.color_palette("cubehelix", 10))
```

In [ ]:
```
sns.palplot(sns.cubehelix_palette(10))
```

creating a density plot to showcase palette

In [ ]:
```
sample_cmap=sns.cubehelix_palette(light=1,as_cmap=True)
sample_cmap
```

In [ ]:
```
# kde plot represents probability distribution of a continuous variable
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]], size=300).T
sns.kdeplot(x=x,y=y,cmap=sample_cmap,fill=True)
```

using color palette in swarm plot

In [ ]:
```
sns.set_style('whitegrid')
tips=sns.load_dataset('tips')
sns.swarmplot(x="day",y='total_bill',data=tips,palette='viridis')
```

Heat Map

A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors.

In [ ]:
```
x = np.array([[1,2,3,4],[2,3,4,1],[5,4,2,1],[6,7,8,5]])
sns.heatmap(x)
x
```

In [ ]:
```
sns.set()
flights = sns.load_dataset("flights")
flights = flights.pivot_table(index="year",columns='month',values="passengers"
sns.heatmap(flights)
```

Cluster Map

Plot a matrix dataset as a hierarchically-clustered heatmap.

In [ ]:
```
species = iris.pop("species")
sns.clustermap(iris)
```

In [ ]:
```
sns.clustermap(iris, cmap="mako", robust=True)
```

In [ ]:
```
```