APSSDC

Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

(https://apssdc.in)

Day12 Data Analysis Using Python

Data Visualization using Seaborn

Visualization Packages

- Matplotlib
- seaborn
- Bokeh
- Plotly

Visualization Tools

- Tableau
- Power BI
- SAS

Data Visualization using Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics
- Seaborn is a library for making statistical graphics in Python
- Applications:
  - used in visualising data in Machine learning, data Science
  - statistical aggregation to produce informative plots

Day12 Objectives

- Using Seaborn Styles
- Categorical scatterplots:
  - stripplot() (with kind="strip"; the default)
  - swarmplot() (with kind="swarm")
- Categorical distribution plots:
  - boxplot() (with kind="box")
  - violinplot() (with kind="violin")
- Joint plot
- Regression Plots
- Creating heatmaps

- Creating pairplots

In [1]:

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

In [2]:

```
sns.get_dataset_names()
```

```
C:\Users\Jesus\anaconda3\lib\site-packages\seaborn\utils.py:384: GuessedAtPa
rserWarning: No parser was explicitly specified, so I'm using the best avail
able HTML parser for this system ("lxml"). This usually isn't a problem, but
if you run this code on another system, or in a different virtual environmen
t, it may use a different parser and behave differently.

The code that caused this warning is on line 384 of the file C:\Users\Jesus
\anaconda3\lib\site-packages\seaborn\utils.py. To get rid of this warning, p
ass the additional argument 'features="lxml"' to the BeautifulSoup construct
or.

  gh_list = BeautifulSoup(http)
```

Out[2]:

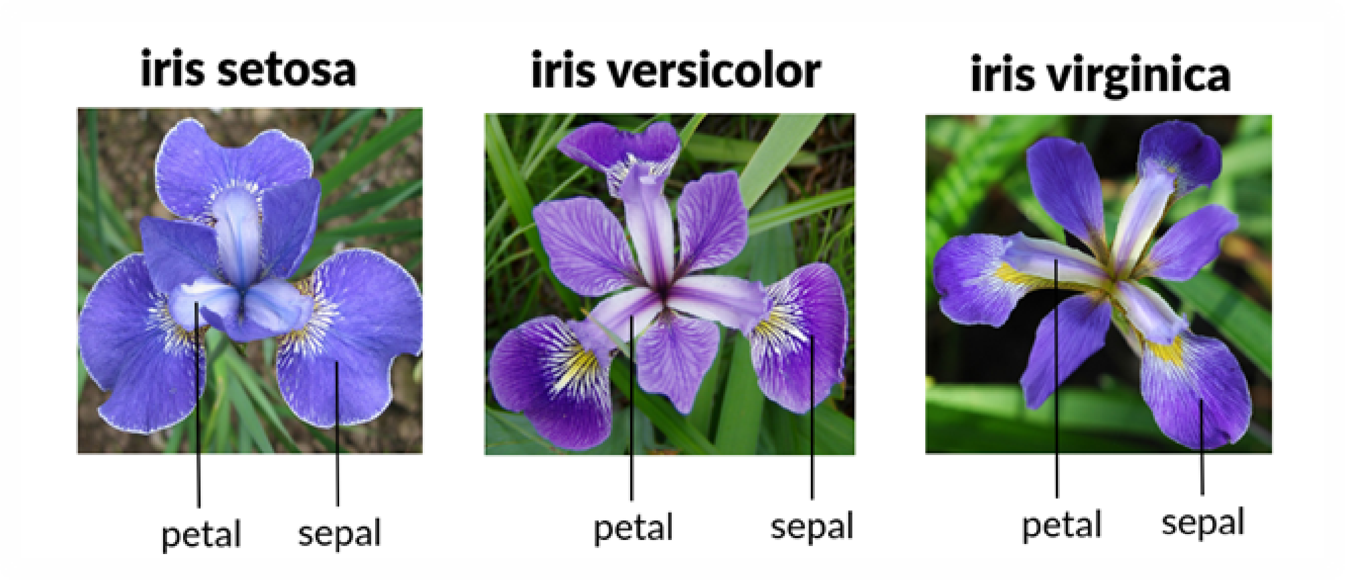
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

In [3]:

```
iris = sns.load_dataset("iris")   ### accesing data sets from seaborn
iris.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa



The columns in this dataset are:

- **SepalLengthCm:** Length of the sepal (in cm)
- **SepalWidthCm:** Width of the sepal (in cm)
- **PetalLengthCm:** Length of the petal (in cm)
- **PetalWidthCm:** Width of the petal (in cm)
- **Species:** Species name

In [4]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:

```
iris.shape
```

Out[5]:

```
(150, 5)
```

In [6]:

```
iris['species'].value_counts()
```

Out[6]:

```
setosa        50
virginica     50
versicolor    50
Name: species, dtype: int64
```

Categorical Scatter Plots

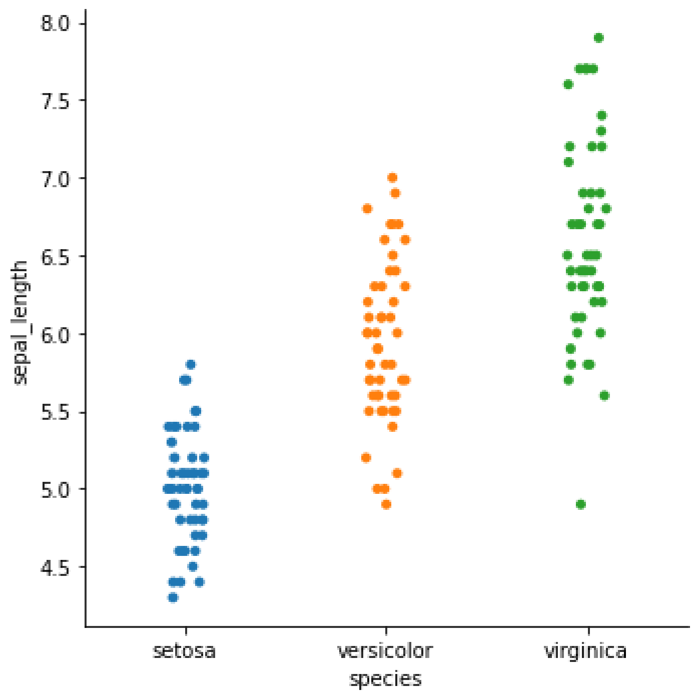
- Features/IV - PL,SL,PW,PL
- Target/OV - Setosa, Virginica, versicolor

In [7]:

```
sns.catplot(x = 'species', y = 'sepal_length', data = iris)
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x2bcde193df0>
```

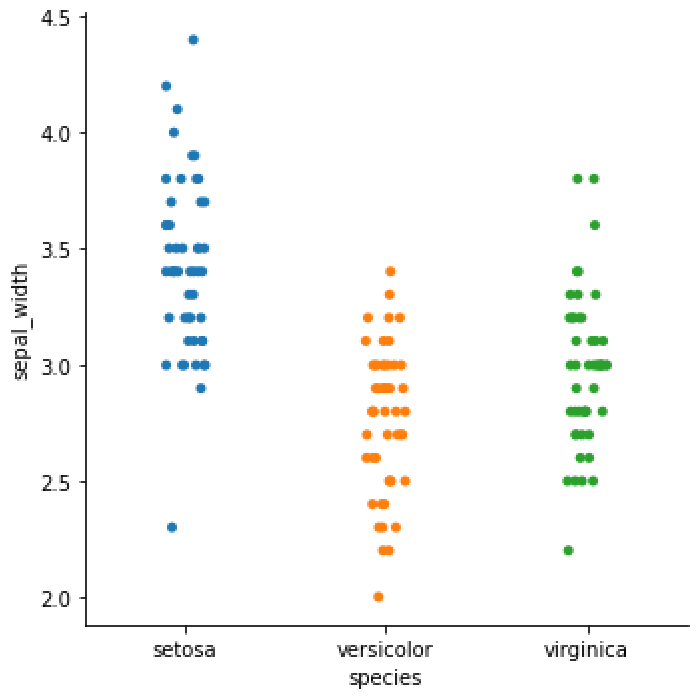


In [8]:

```python
sns.catplot(x = 'species', y = 'sepal_width', data = iris)
```

Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x2bce027b670>
```



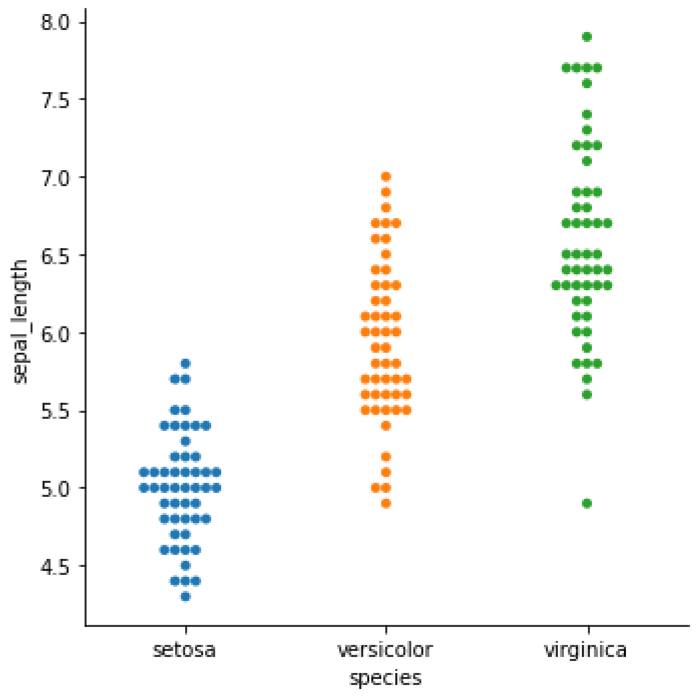
Swarm Plot - adjust the points along the categorical axis using some algorithm that preventing the overlaping of data

In [9]:

```
sns.catplot(x = 'species', y = 'sepal_length', data = iris, kind = 'swarm')
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x2bce0297cd0>
```

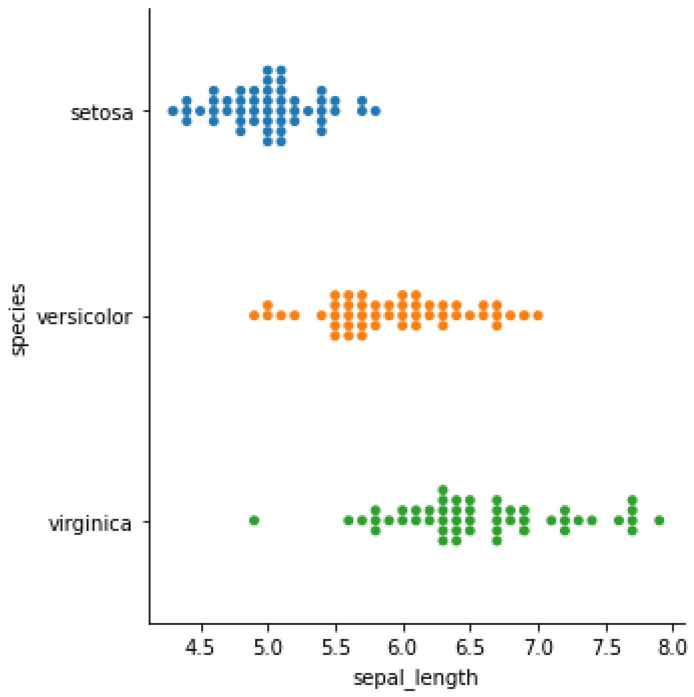


In [10]:

```
sns.catplot(y = 'species', x = 'sepal_length', data = iris, kind = 'swarm')
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x2bcde1c1250>
```

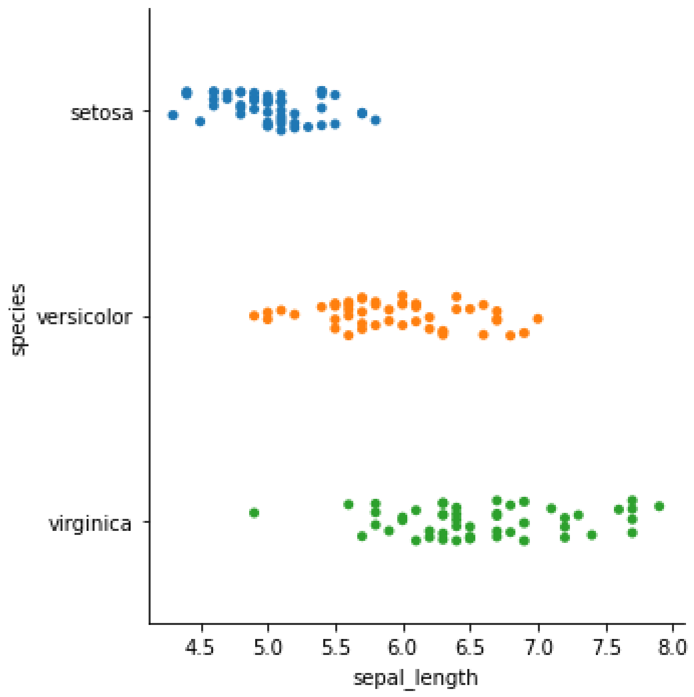


In [11]:

```
sns.catplot(y = 'species', x = 'sepal_length', data = iris, kind = 'strip')
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x2bcde1939a0>
```



Caterorical Distribution Data

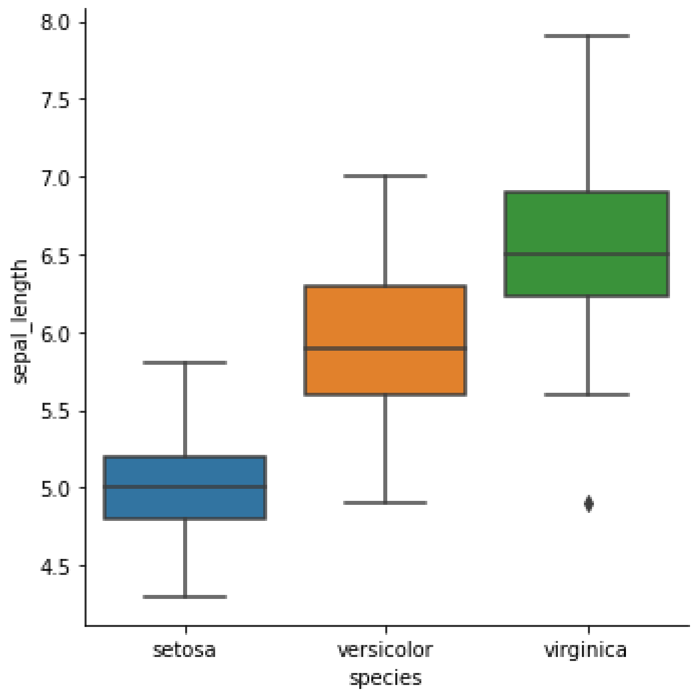
Box Plot

In [13]:

```
sns.catplot(x = 'species', y = 'sepal_length', data = iris, kind = 'box')
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x2bce1176a60>
```

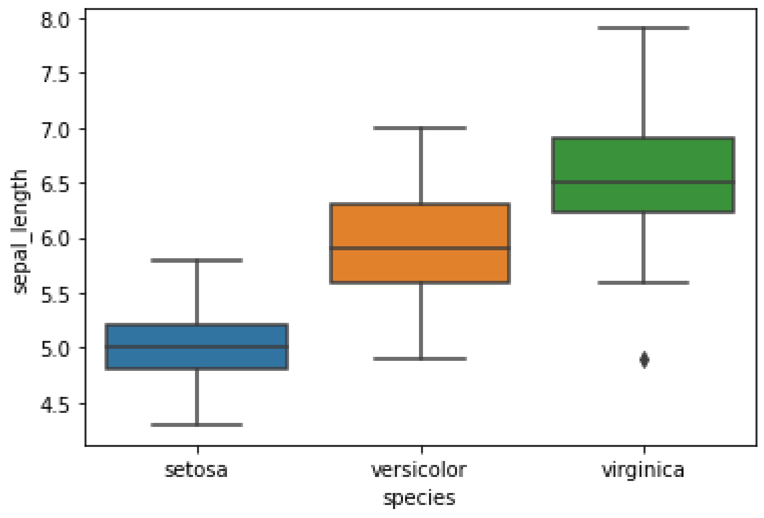


In [15]:

```
sns.boxplot(x = 'species', y = 'sepal_length', data = iris)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bce124eca0>
```

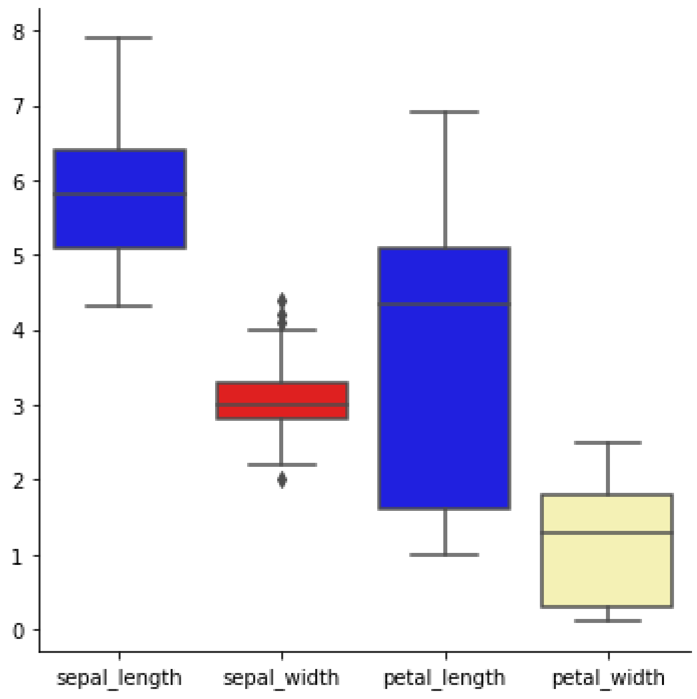


In [17]:

```
sns.catplot(data = iris, orient = 'v', palette = ['b','r','b','#fffaab'], kind = 'box')
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x2bce145d490>
```



Violin Plots

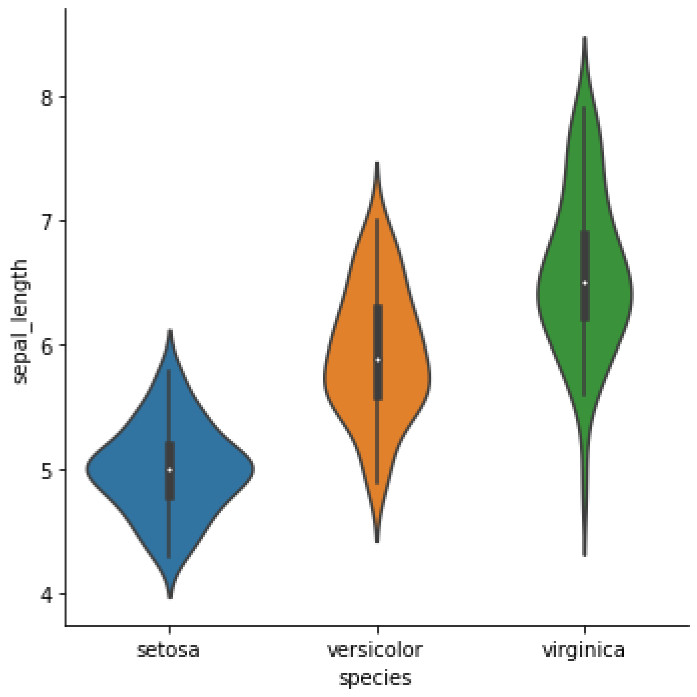
it is the combination of Box plot as well as the KDE (Kernal Density estimation)

In [18]:

```
sns.catplot(x = 'species', y = 'sepal_length', data = iris, kind = 'violin')
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x2bce14dbe80>
```



Joint Plot

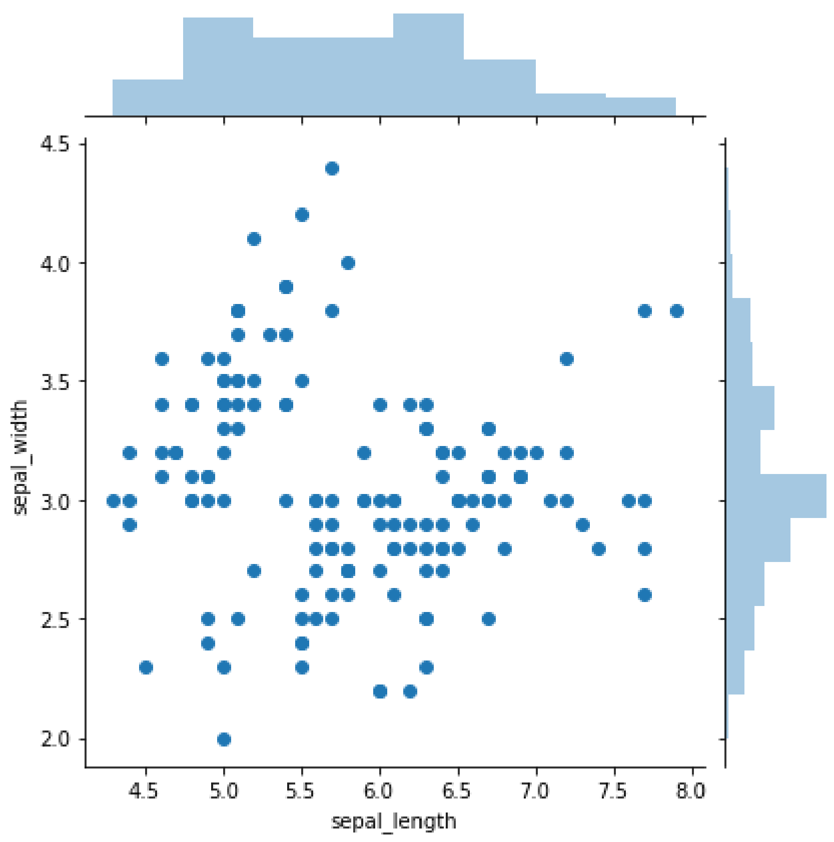
Combine mutiple kinds plots for getting the insites from the data

In [25]:

```
sns.jointplot(x = 'sepal_length', y='sepal_width', data= iris)
plt.show()
```

Out[25]:

```
<seaborn.axisgrid.JointGrid at 0x2bce4985100>
```



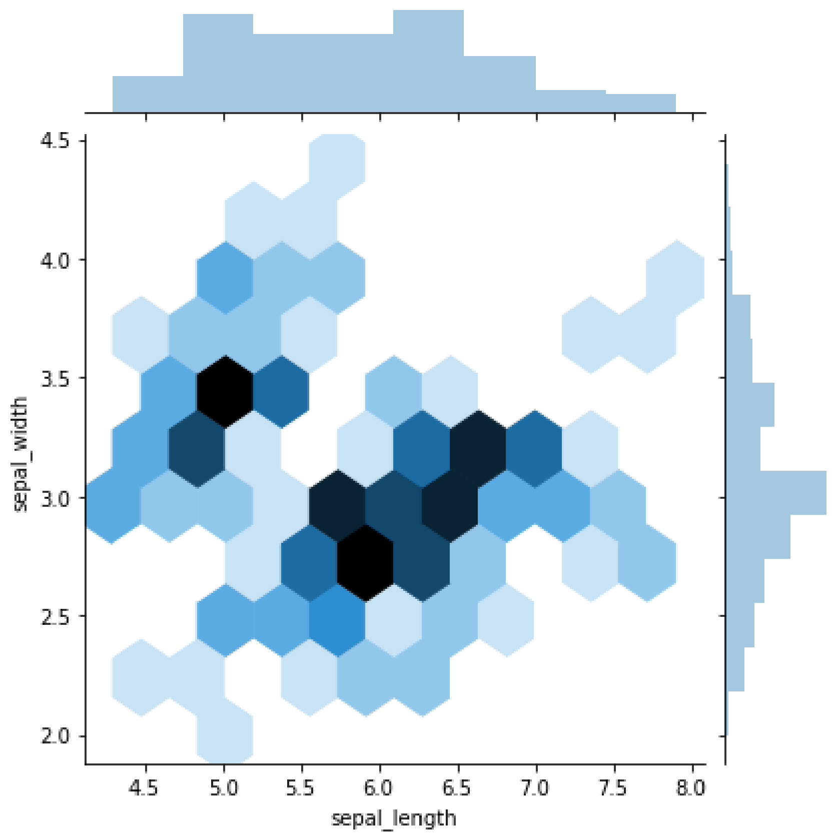
In [24]:

```
help(sns.jointplot)
```

```
Help on function jointplot in module seaborn.axisgrid:

jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, hei
ght=6, ratio=5, space=0.2, dropna=True, xlim=None, ylim=None, joint_kws=No
ne, marginal_kws=None, annot_kws=None, **kwargs)
    Draw a plot of two variables with bivariate and univariate graphs.

    This function provides a convenient interface to the :class:`JointGrid
`
    class, with several canned plot kinds. This is intended to be a fairly
    lightweight wrapper; if you need more flexibility, you should use
    :class:`JointGrid` directly.

    Parameters
    ----------
    x, y : strings or vectors
        Data or names of variables in ``data``.
    data : DataFrame, optional
        DataFrame when ``x`` and ``y`` are variable names.
```

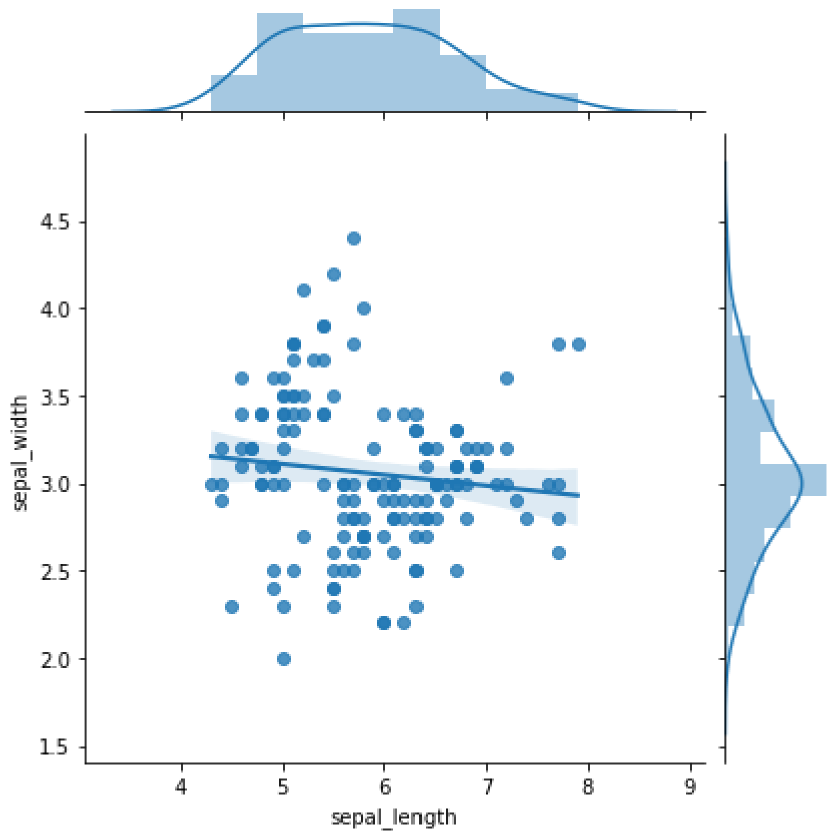
In [26]:

```
sns.jointplot(x = 'sepal_length', y='sepal_width', kind = 'hex',data= iris)
plt.show()
```



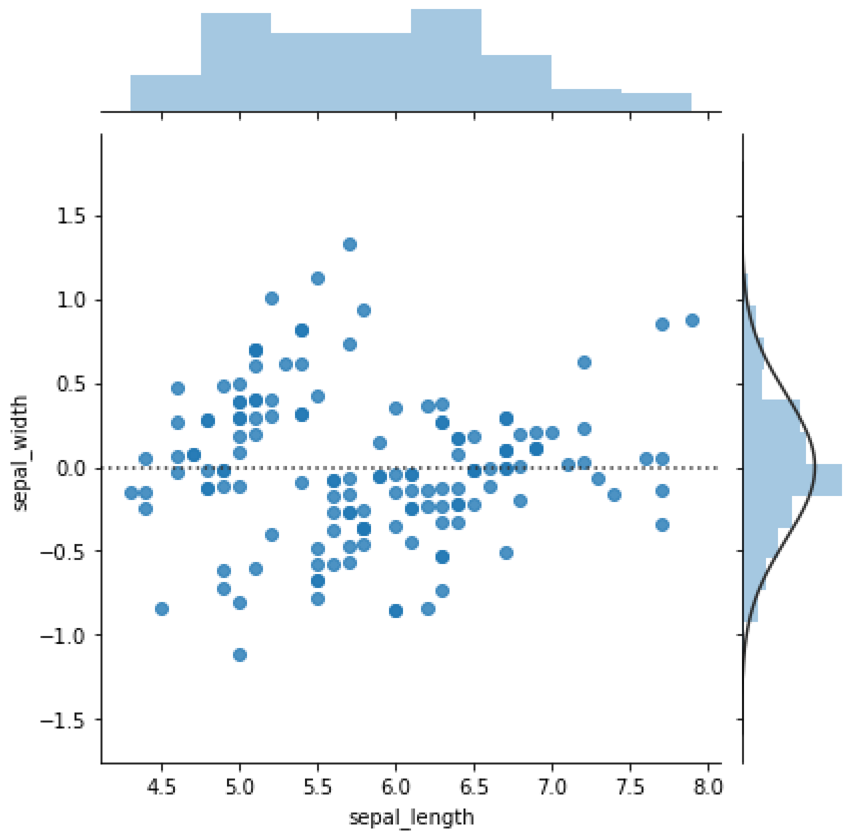
In [27]:

```
sns.jointplot(x = 'sepal_length', y='sepal_width', kind = 'reg',data= iris)
plt.show()
```



In [28]:

```
sns.jointplot(x = 'sepal_length', y='sepal_width', kind = 'resid',data= iris)
plt.show()
```

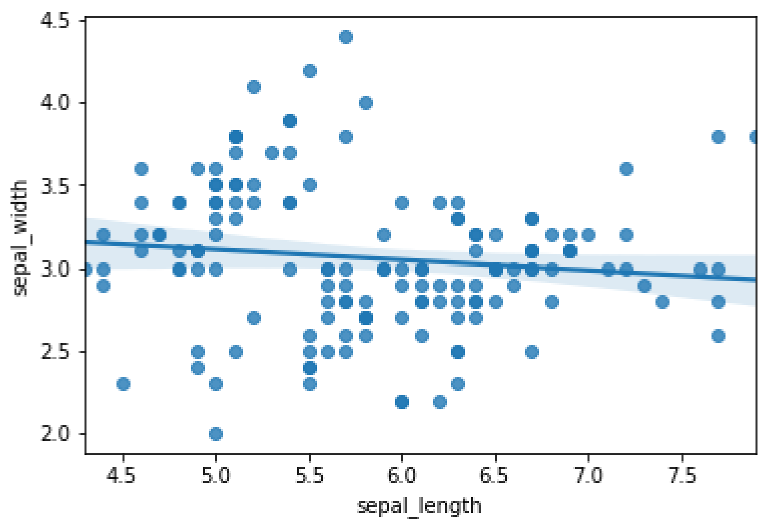


In [29]:

```
sns.regplot(x = 'sepal_length', y='sepal_width', data= iris)
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bce4dc62e0>
```



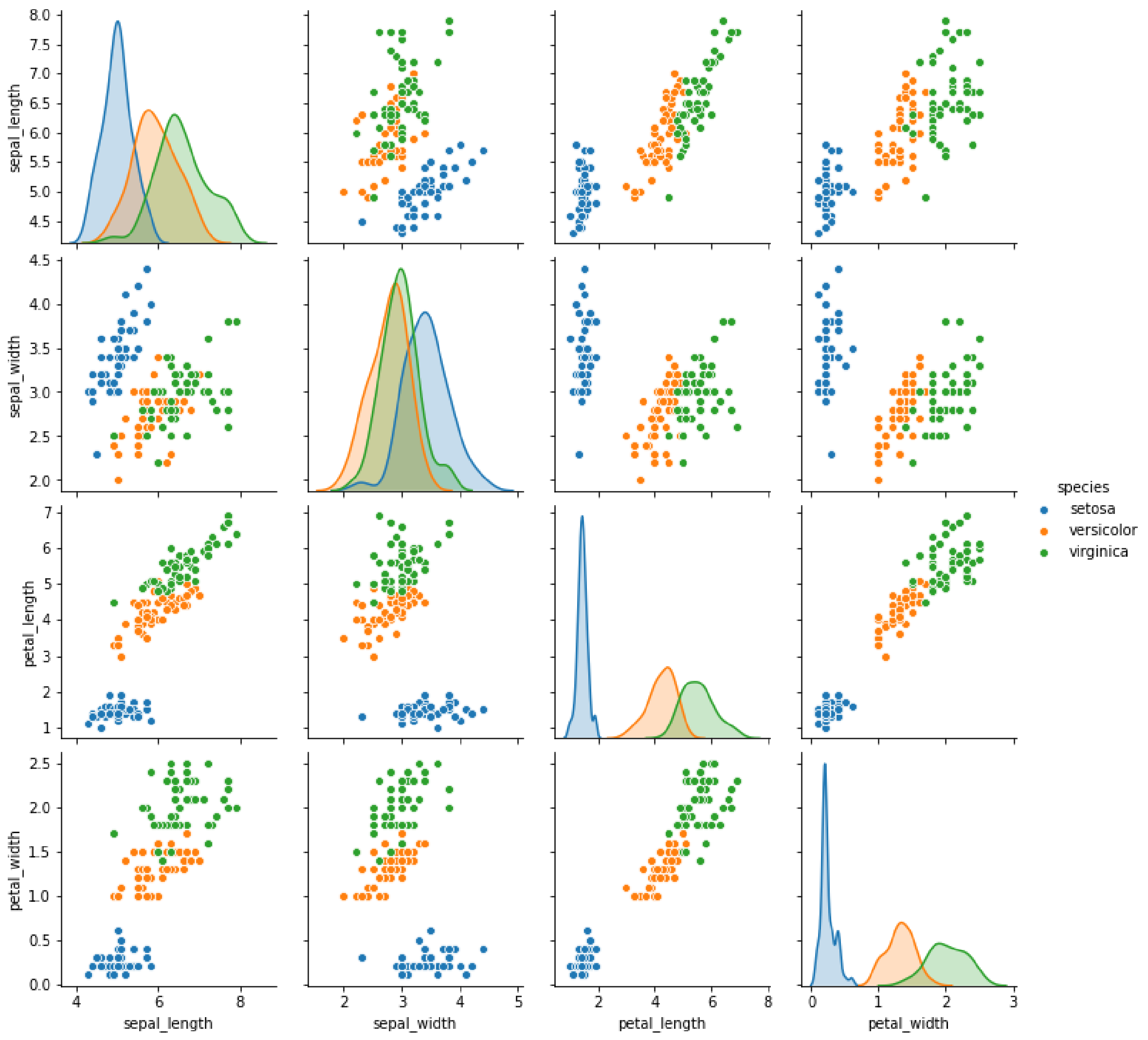
Pair Plot

In [30]:

```
sns.pairplot(data = iris, hue = 'species')
```

Out[30]:

```
<seaborn.axisgrid.PairGrid at 0x2bce4dd6fd0>
```

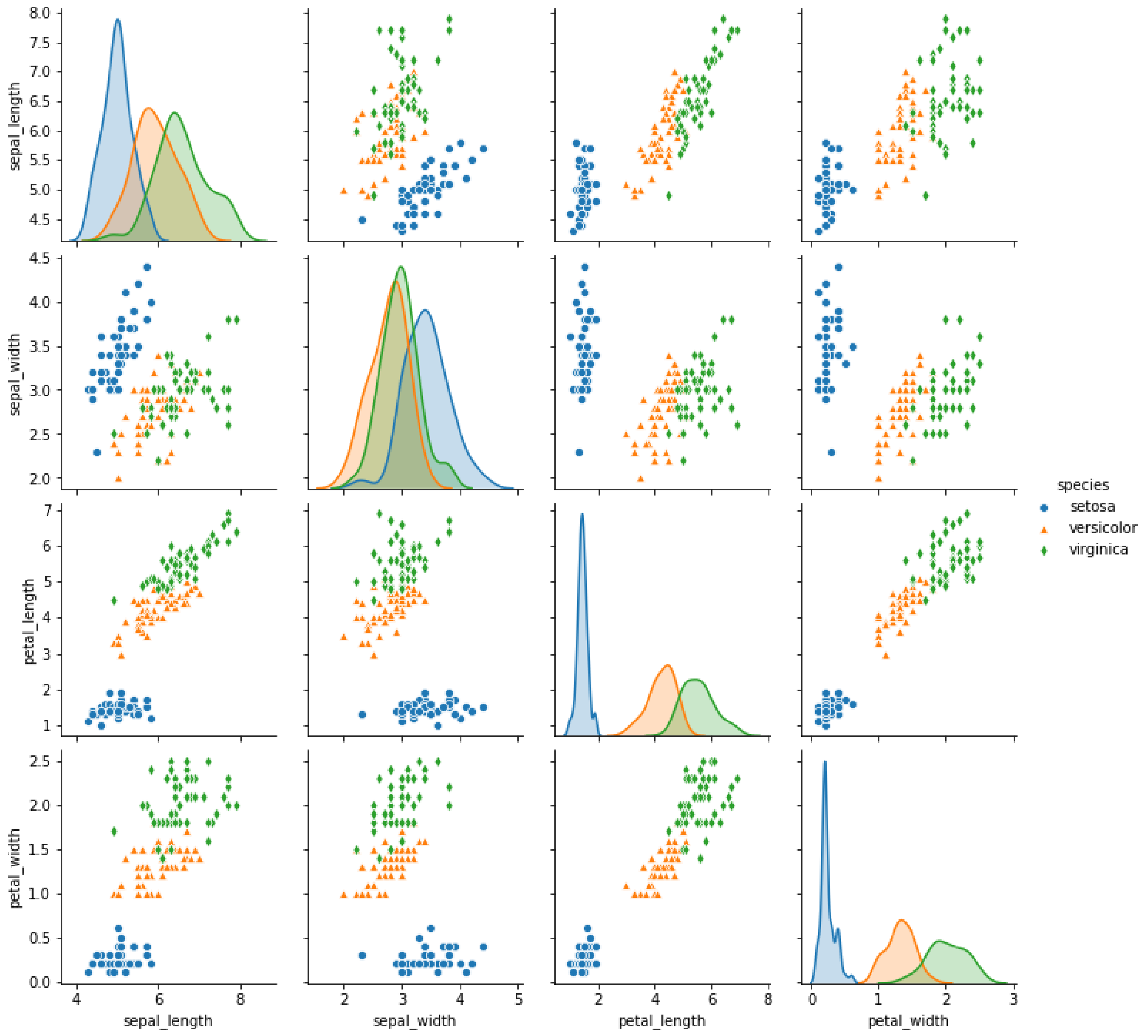


In [32]:

```
sns.pairplot(data = iris, hue = 'species', markers = ['o','^','d'])
```

Out[32]:

```
<seaborn.axisgrid.PairGrid at 0x2bce4bd1130>
```



In [33]:

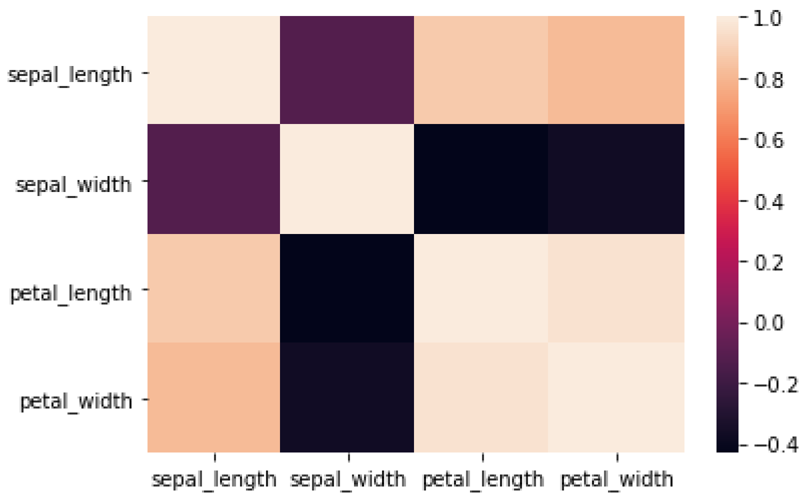
```
corr = iris.corr()
```

In [34]:

```
sns.heatmap(corr)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bce6e36580>
```

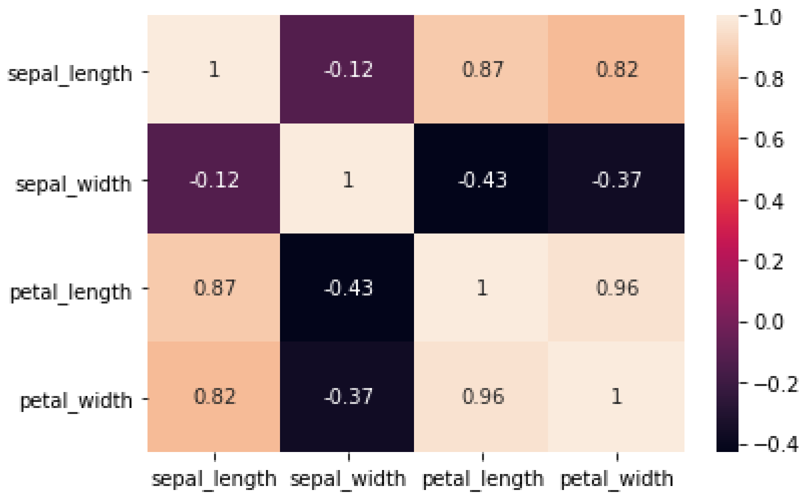


In [35]:

```
sns.heatmap(corr, annot = True)
```

Out[35]:

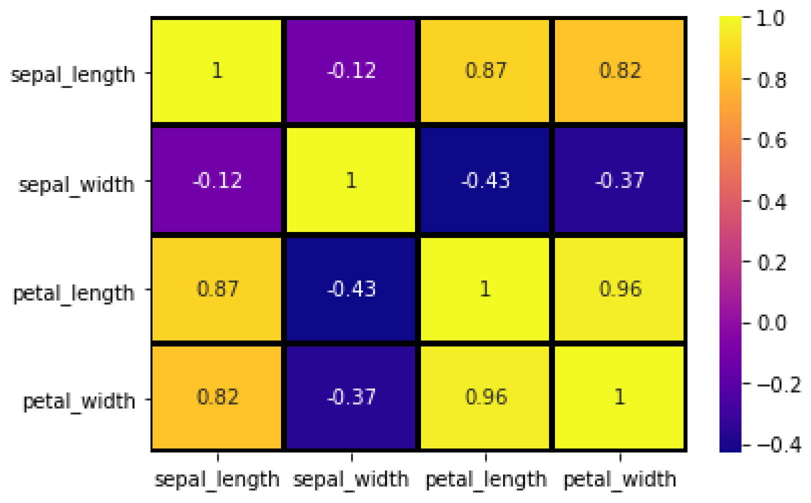
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bce711f8e0>
```



```
In [37]:
```

```python
sns.heatmap(corr, annot = True, cmap = 'plasma', linecolor = 'black', linewidth = 2)
```

```
Out[37]:

<matplotlib.axes._subplots.AxesSubplot at 0x2bce7124850>
```



```
In [36]:
```

```python
help(sns.heatmap)
```

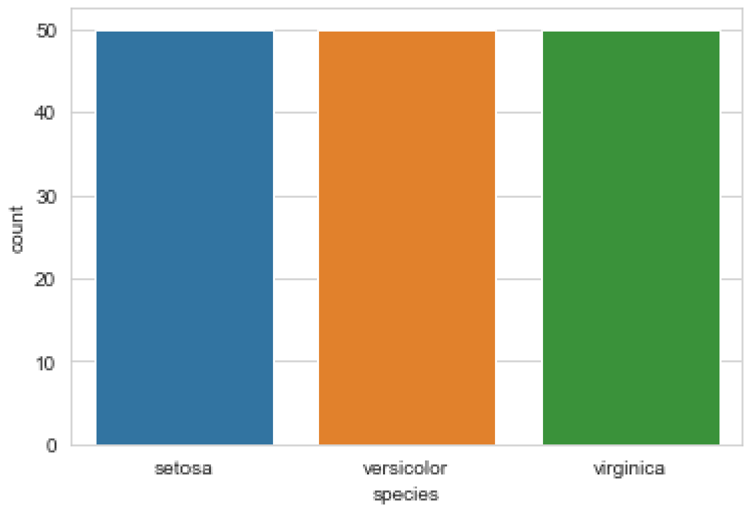
```
Help on function heatmap in module seaborn.matrix:

heatmap(data, vmin=None, vmax=None, cmap=None, center=None, robust=False,
annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cb
ar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yt
icklabels='auto', mask=None, ax=None, **kwargs)
    Plot rectangular data as a color-encoded matrix.

    This is an Axes-level function and will draw the heatmap into the
    currently-active Axes if none is provided to the ``ax`` argument.  Par
t of
    this Axes space will be taken and used to plot a colormap, unless ``cb
ar``
    is False or a separate Axes is provided to ``cbar_ax``.

    Parameters
    ----------
    data : rectangular dataset
        2D dataset that can be coerced into an ndarray. If a Pandas DataFr
```

In [39]:

```
sns.set_style('whitegrid')
sns.countplot(iris['species'])
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bce7292a90>
```


count
setosa
versicolor
virginica
species