25 (1) Linear Algebra [Vector , Scalar, Tensors, Matrix, Gradiant, Eigen Values and Vectors]

In [10]:
```
# pro 1(25)
import numpy as n
import tensorflow as t

s=50
print("Scalar : ",s)

m=n.array([[0, 2],[2, 3]])
v=n.array([[0, 2]])
print("Matrix:",m)
print("Vector:",v)

print("Tensor:")
fill_2d = t.fill([3, 3],4, '2d')
fill_string = t.fill([2, 2], "str", 'fill_tensor_string')
print("Numerics:",fill_2d)
print("String:",fill_string)
g=n.gradient(m)
print("Gradient:",g)

w,v = n.linalg.eig(m)
mat_norm = n.linalg.norm(m)
print("Eigen values:",w)
print("Eigen vectors:",v)
print("Matrix norm:", mat_norm)
```

```
Scalar :  50
Matrix: [[0 2]
 [2 3]]
Vector: [[0 2]]
Tensor:
Numerics: tf.Tensor(
[[4 4 4]
 [4 4 4]
 [4 4 4]], shape=(3, 3), dtype=int32)
String: tf.Tensor(
[[b'str' b'str']
 [b'str' b'str']], shape=(2, 2), dtype=string)
Gradient: [array([[2., 1.],
       [2., 1.]]), array([[2., 2.],
       [1., 1.]])]
Eigen values: [-1.  4.]
Eigen vectors: [[-0.89442719 -0.4472136 ]
 [ 0.4472136  -0.89442719]]
Matrix norm: 4.123105625617661
```

26 (2) Covariance and Co-relation

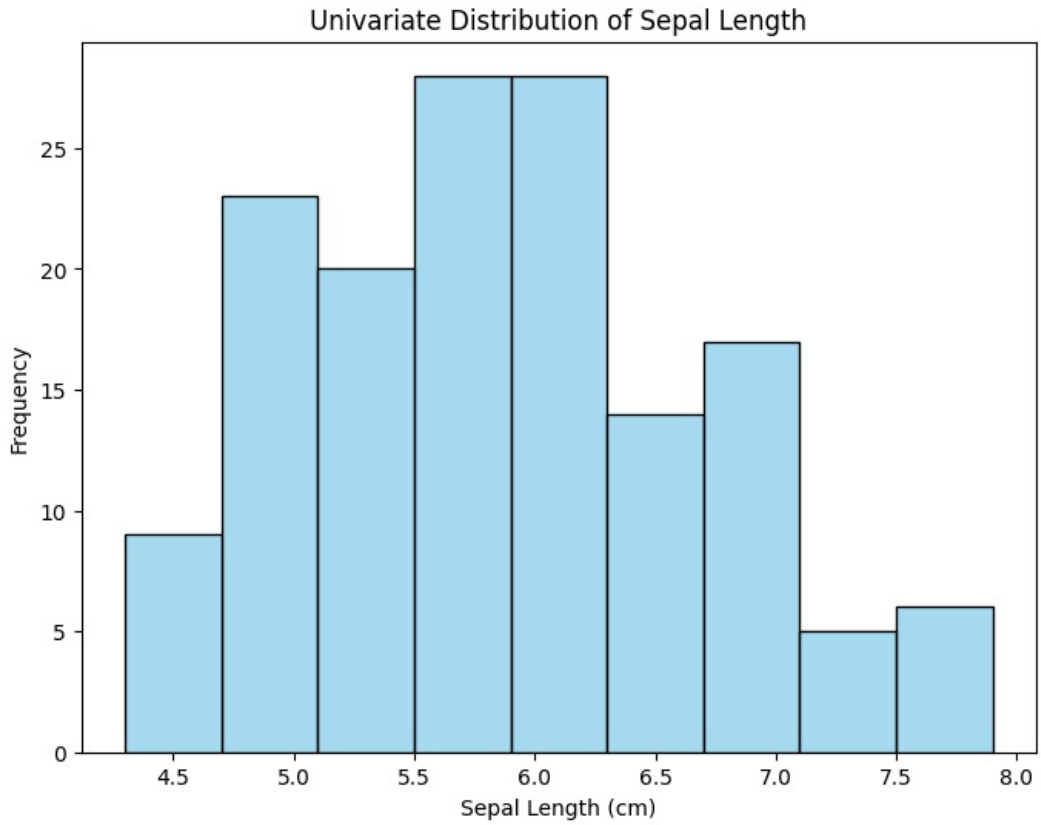
In [21]:
```
# pro 2 (26)
import pandas as pd
from sklearn import datasets

iris = datasets.load_iris()

df = pd.DataFrame(iris.data, columns=["sepal_length", "sepal_width",
"petal_length", "petal_width"])

df["class"] = iris.target

cov=df.iloc[:, 0:4].cov()
cor=df.iloc[:, 0:4].corr()

print("Covariance:\n",cov)
print("\nCorelation:\n",cor)
```

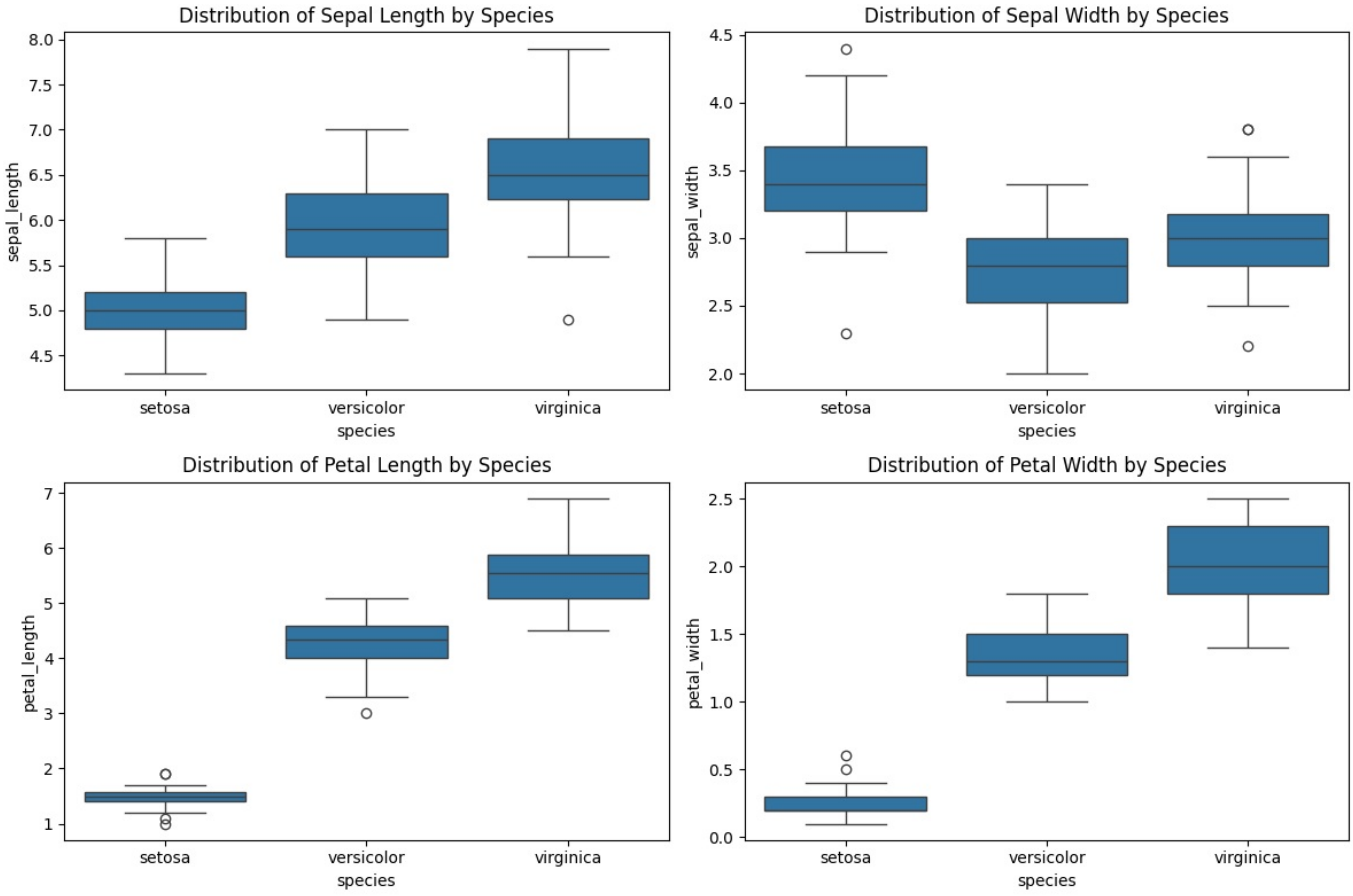
```
Covariance:
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      0.685694    -0.042434      1.274315     0.516271
sepal_width      -0.042434     0.189979     -0.329656    -0.121639
petal_length      1.274315    -0.329656      3.116278     1.295609
petal_width       0.516271    -0.121639      1.295609     0.581006

Corelation:
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      1.000000    -0.117570      0.871754     0.817941
sepal_width      -0.117570     1.000000     -0.428440    -0.366126
petal_length      0.871754    -0.428440      1.000000     0.962865
petal_width       0.817941    -0.366126      0.962865     1.000000
```

27 (3) Univariate & Multivariate Distrubution Plot

In [25]:

```python
# pro 3 (27)
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

#Univariate
sepal_length_data = iris["sepal_length"]

plt.figure(figsize=(8, 6))

sns.histplot(sepal_length_data, color="skyblue")

plt.xlabel("Sepal Length (cm)")
plt.ylabel("Frequency")
plt.title("Univariate Distribution of Sepal Length")

plt.show()

#Multivariate

# Create subplots for each numeric variable
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

# Plot sepal length distribution by species
sns.boxplot(x="species", y="sepal_length", data=iris, ax=axes[0, 0])
axes[0, 0].set_title("Distribution of Sepal Length by Species")

# Plot sepal width distribution by species
sns.boxplot(x="species", y="sepal_width", data=iris, ax=axes[0, 1])
axes[0, 1].set_title("Distribution of Sepal Width by Species")

# Plot petal length distribution by species
sns.boxplot(x="species", y="petal_length", data=iris, ax=axes[1, 0])
axes[1, 0].set_title("Distribution of Petal Length by Species")

# Plot petal width distribution by species
sns.boxplot(x="species", y="petal_width", data=iris, ax=axes[1, 1])
axes[1, 1].set_title("Distribution of Petal Width by Species")

plt.tight_layout()
plt.show()
```



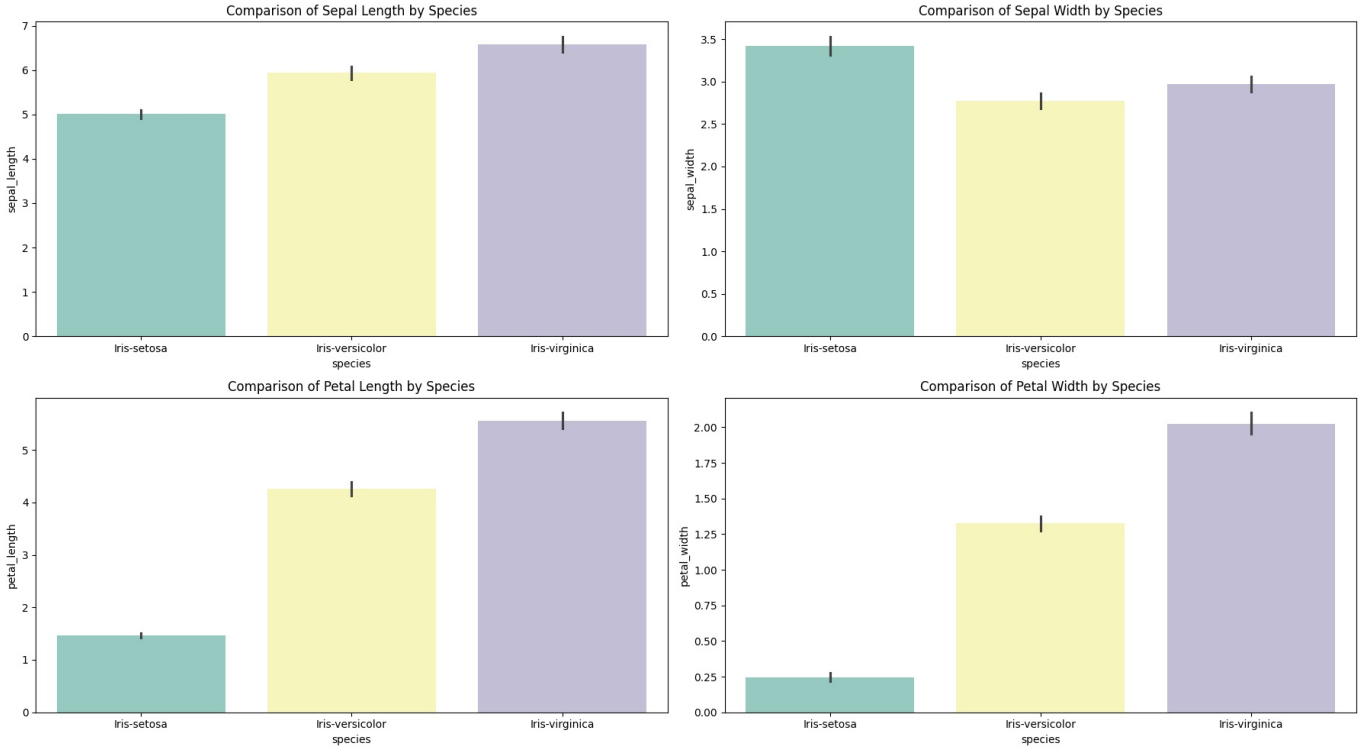


28 (4) Univariate & Multivariate Comparision Plots

In [8]:
```python
# pro 4 (28)
import seaborn as sns
from matplotlib import pyplot as plt
import pandas as p

iris = p.read_csv("IRIS.csv")

# Set the figure size
plt.figure(figsize=(18, 10))

# Create grouped bar plots for sepal length, sepal width, petal length, and petal width by species
```

```
plt.subplot(2, 2, 1)
sns.barplot(x="species", y="sepal_length", data=iris, palette="Set3")
plt.title("Comparison of Sepal Length by Species")

plt.subplot(2, 2, 2)
sns.barplot(x="species", y="sepal_width", data=iris, palette="Set3")
plt.title("Comparison of Sepal Width by Species")

plt.subplot(2, 2, 3)
sns.barplot(x="species", y="petal_length", data=iris, palette="Set3")
plt.title("Comparison of Petal Length by Species")

plt.subplot(2, 2, 4)
sns.barplot(x="species", y="petal_width", data=iris, palette="Set3")
plt.title("Comparison of Petal Width by Species")

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

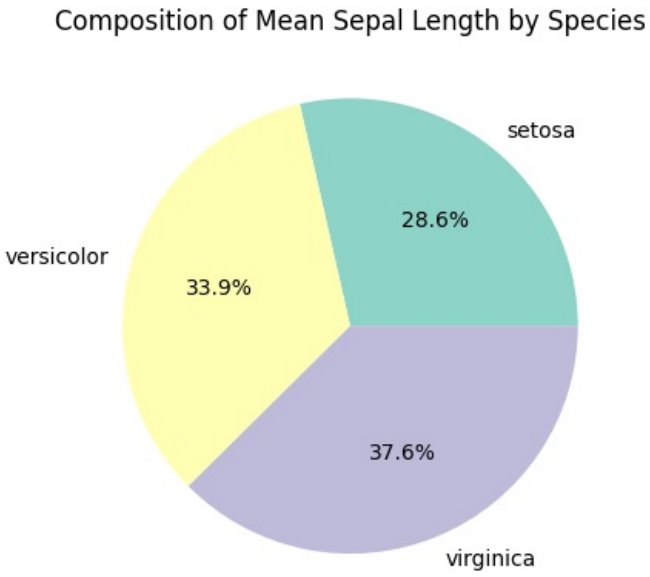
```
C:\Users\sandy\AppData\Local\Temp\ipykernel_20468\1856293275.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="species", y="sepal_length", data=iris, palette="Set3")
C:\Users\sandy\AppData\Local\Temp\ipykernel_20468\1856293275.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="species", y="sepal_width", data=iris, palette="Set3")
C:\Users\sandy\AppData\Local\Temp\ipykernel_20468\1856293275.py:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="species", y="petal_length", data=iris, palette="Set3")
C:\Users\sandy\AppData\Local\Temp\ipykernel_20468\1856293275.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="species", y="petal_width", data=iris, palette="Set3")
```

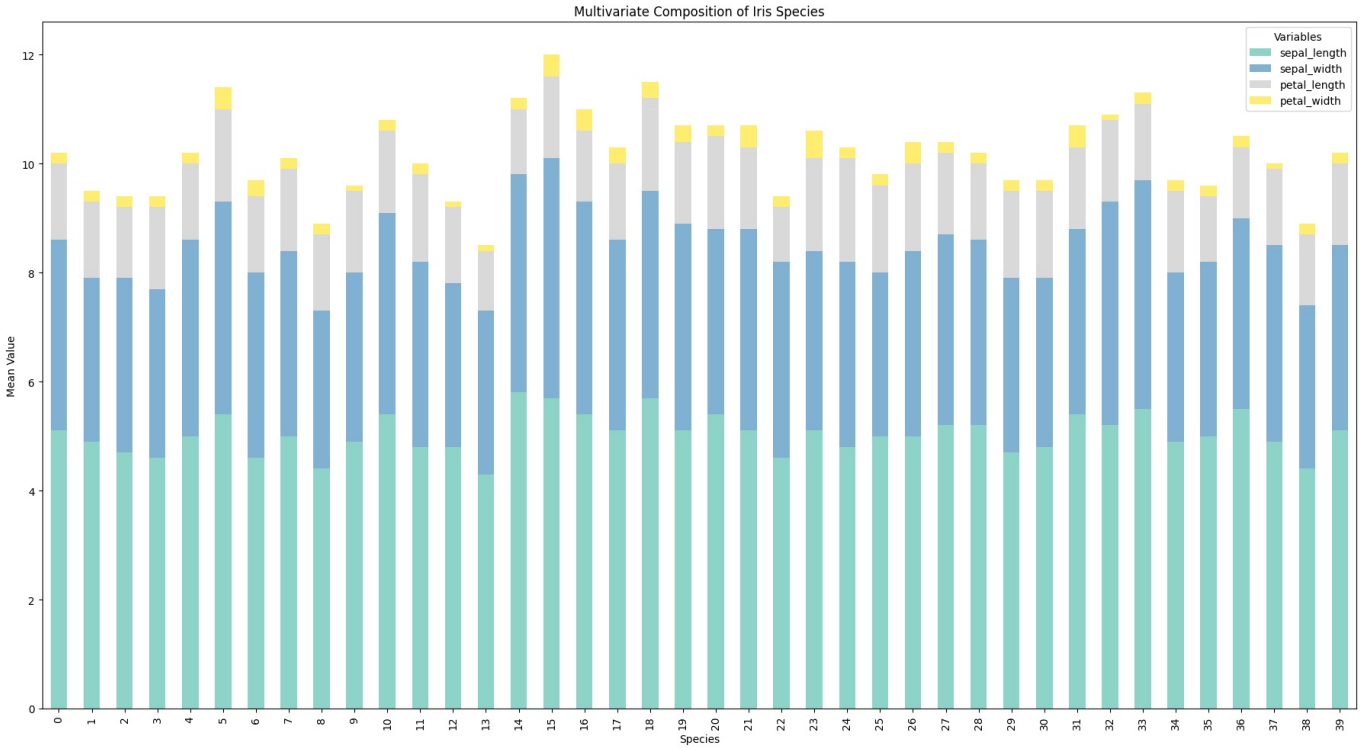


29 (5) Univariate & Multivariate Composition Plot

In [58]:
```
# program 5 (29)
import seaborn as sns
```

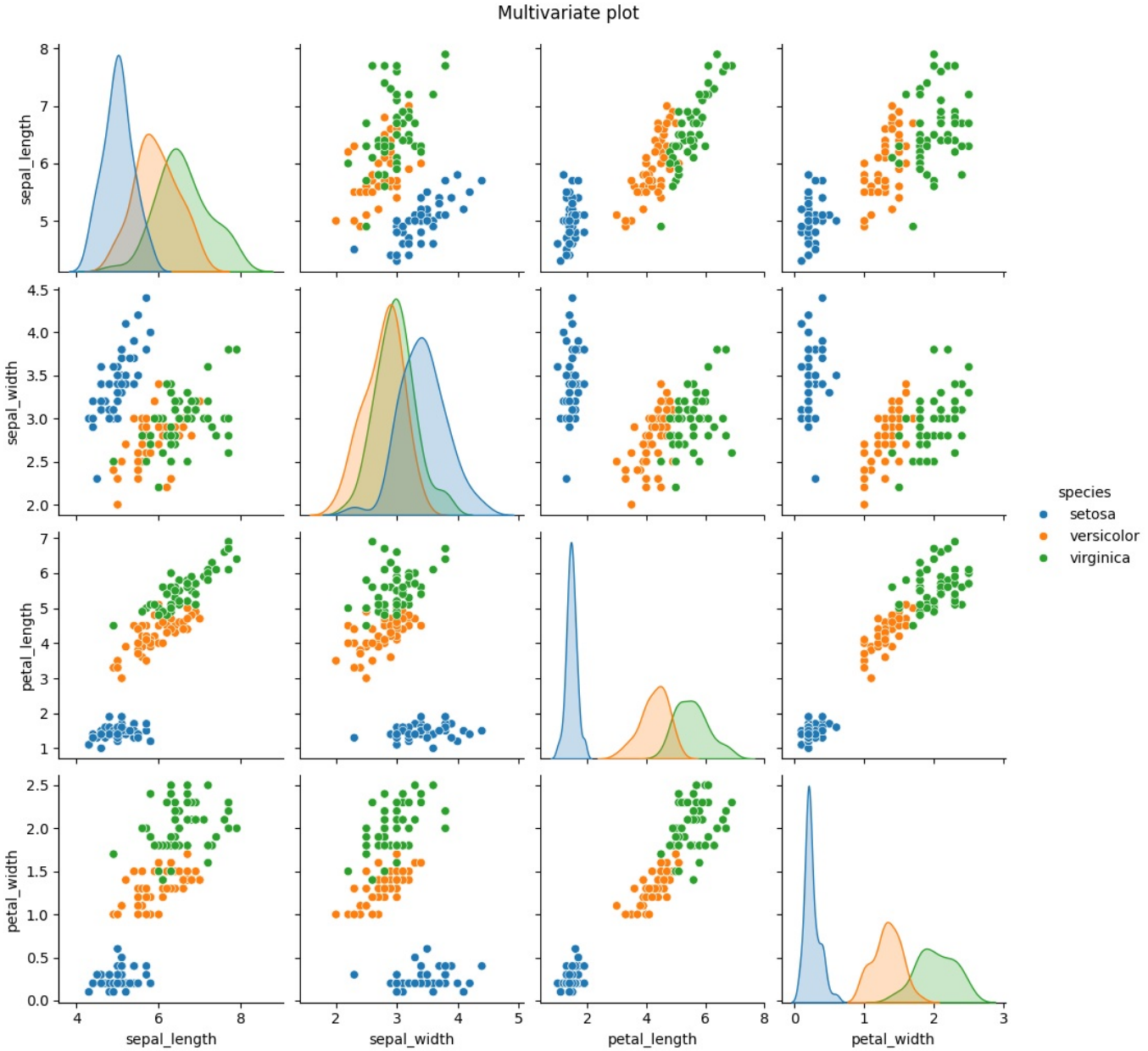
```
import matplotlib.pyplot as plt
# Load the Iris dataset
iris = sns.load_dataset("iris")

#univariate
sl= iris.groupby("species")["sepal_length"].mean()
plt.pie(sl, labels=sl.index, autopct='%1.1f%%',
colors=sns.color_palette("Set3"))
plt.title("Composition of Mean Sepal Length by Species")
plt.show()

#Multivariate
#Reducing the dataset to count of 40
iris = sns.load_dataset("iris").head(40)
# Group data by species and calculate the mean of numeric variables
species_data = iris.groupby("species")[["sepal_length", "sepal_width",
"petal_length", "petal_width"]]

# Create a stacked bar chart
species_data.plot(kind="bar", stacked=True, colormap="Set3", figsize=(18, 10))

# Set labels and title
plt.title("Multivariate Composition of Iris Species")
plt.xlabel("Species")
plt.ylabel("Mean Value")

# Show the plot
plt.legend(title="Variables", loc="upper right")
plt.tight_layout()
plt.show()
```





30 (6) Multivariate Relationship Plot

In [61]:
```
# pro 6 (30)
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

df = sns.load_dataset('iris')

# plt.figure(figsize=(24,20))
sns.pairplot(df, hue="species")
plt.suptitle('Multivariate plot',y=1.02)
plt.show()
```



In [ ]: