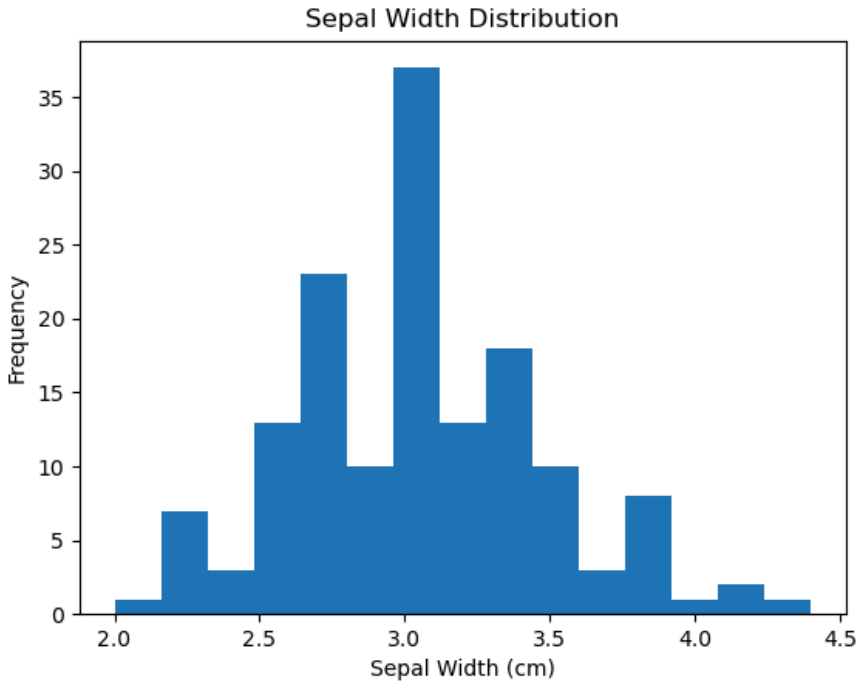
Pooja Rejith
Foundations in Data Science
1 February 2026

Week 3 Assignment

1. Using the iris dataset…
    a. Make a histogram of the variable Sepal.Width.
        i.

```
(base) pooja@Poojas-Air ~ % python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 08:22:19) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Cmd click to launch VS Code Native REPL
>>> import sklearn
>>> from sklearn import datasets
>>> import pandas as pd
>>> iris = datasets.load_iris()
>>> iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
>>> data = { "weight": [4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14, 4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89,
4.32, 4.69, 6.31, 5.12, 5.54, 5.50, 5.37, 5.29, 4.92, 6.15, 5.80, 5.26], "group": ["ctrl"] * 10 + ["trt1"] * 10 + ["trt2"] * 10}
>>> PlantGrowth = pd.DataFrame(data)
>>> iris_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
>>> import matplotlib.pyplot as plt
>>> plt.hist(iris_df['sepal width (cm)'], bins=15)
(array([ 1.,  7.,  3., 13., 23., 10., 37., 13., 18., 10.,  3.,  8.,  1.,
        2.,  1.]), array([2.  , 2.16, 2.32, 2.48, 2.64, 2.8 , 2.96, 3.12, 3.28, 3.44, 3.6 ,
       3.76, 3.92, 4.08, 4.24, 4.4 ]), <BarContainer object of 15 artists>)
>>> plt.xlabel("Sepal Width (cm)")
Text(0.5, 47.04444444444444, 'Sepal Width (cm)')
>>> plt.ylabel("Frequency")
Text(94.19444444444443, 0.5, 'Frequency')
>>> plt.title("Sepal Width Distribution")
Text(0.5, 1.0, 'Sepal Width Distribution')
>>> plt.show()
```

        ii.



    b. Based on the histogram from #1a, which would you expect to be higher, the mean or the median? Why?

i. The mean should be higher than the median. This histogram is slightly right-skewed, with the tail extending out into values between 4.0 cm and 4.5 cm. These outlier values will raise the overall mean and therefore create the expectation of a higher mean relative to the median.

c. Confirm your answer to #1b by actually finding these values.

i.
```
>>> print('Mean: ' + str(iris_df['sepal width (cm)'].mean()))
Mean: 3.0573333333333337
>>> print('Median: ' + str(iris_df['sepal width (cm)'].median()))
Median: 3.0
```

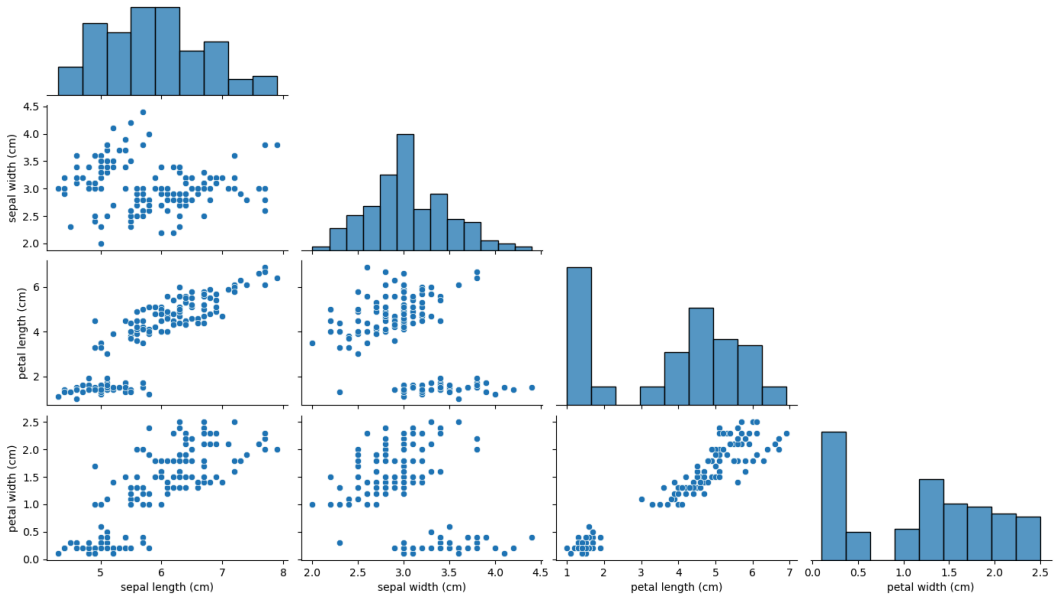
d. Only 27% of the flowers have a Sepal.Width higher than **3.3** cm.

i.
```
>>> import numpy as np
>>> np.percentile(iris_df['sepal width (cm)'], 73)
3.3
```

e. Make scatterplots of each pair of the numerical variables in iris (There should be 6 pairs/plots).

i.
```
>>> import seaborn as sns
>>> sns.pairplot(iris_df, corner=True)
<seaborn.axisgrid.PairGrid object at 0x16c0607a0>
>>> plt.show()
```

ii.



f. Based on #1e, which two variables appear to have the strongest relationship? And which two appear to have the weakest relationship?

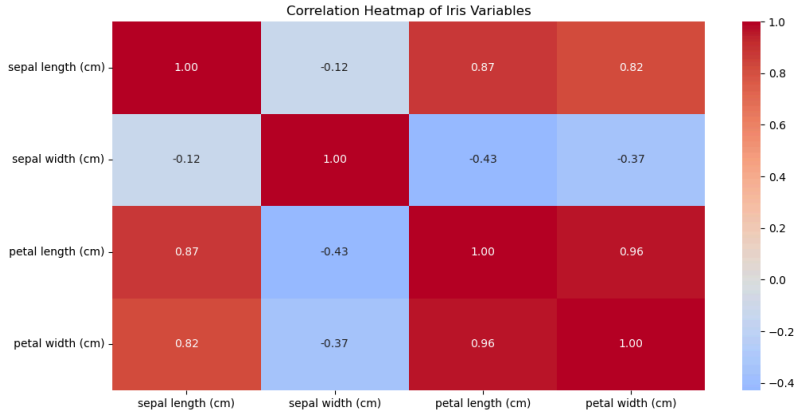
i. Petal Length vs Petal Width seems to have the strongest relationship, as there is a very strong positive correlation prevalent in the scatter plot that depicts their relationship. Meanwhile, Sepal Width vs Sepal Length seems to have the weakest relationship, as there is no correlation that can be deduced based on the scatterplot shown above. I also created a correlation matrix using seaborn to confirm my hypothesis - as shown below, Petal Length vs Petal Width has the highest correlation with a

value of 0.96, and Sepal Width vs Sepal Length has the lowest correlation with a value of -0.12.

ii.

```
>>> corr = iris_df.corr()
>>> plt.figure(figsize=(8, 6))
<Figure size 1600x1200 with 0 Axes>
>>> sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", center=0)
<Axes: >
>>> plt.title("Correlation Heatmap of Iris Variables")
Text(0.5, 1.0, 'Correlation Heatmap of Iris Variables')
>>> plt.show()
```

iii.

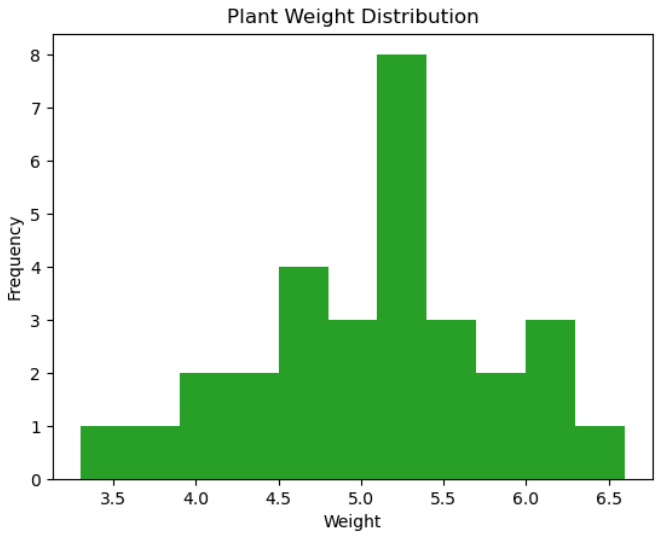


2. Using the PlantGrowth dataset…
    a. Make a histogram of the variable weight with breakpoints (bin edges) at every 0.3 units, starting at 3.3.

i.

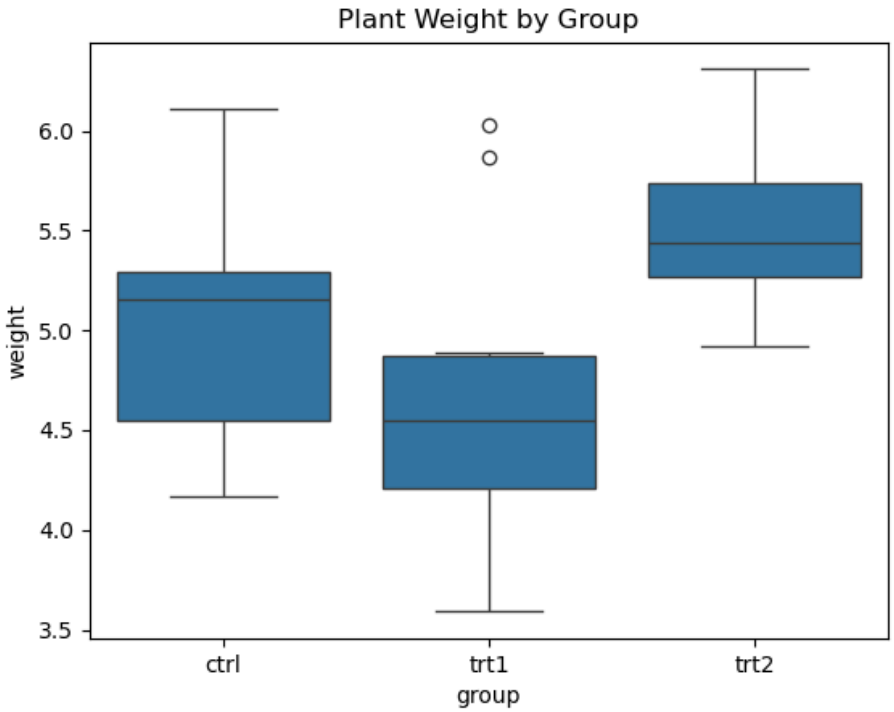
```
>>> bins = np.arange(3.3, PlantGrowth.weight.max() + 0.3, 0.3)
>>> plt.hist(PlantGrowth.weight, bins=bins)
(array([1., 1., 2., 2., 4., 3., 8., 3., 2., 3., 1.]), array([3.3, 3.6, 3.9, 4.2, 4.5, 4.8, 5.1, 5.4, 5.7, 6. , 6.3, 6.6]), <BarC
ontainer object of 11 artists>)
>>> plt.xlabel("Weight")
Text(0.5, 47.04444444444444, 'Weight')
>>> plt.ylabel("Frequency")
Text(111.81944444444443, 0.5, 'Frequency')
>>> plt.title("Plant Weight Distribution")
Text(0.5, 1.0, 'Plant Weight Distribution')
>>> plt.show()
```

ii.



b. Make boxplots of weight separated by group in a single graph.

i.
```
>>> sns.boxplot(x="group", y="weight", data=PlantGrowth)
<Axes: xlabel='group', ylabel='weight'>
>>> plt.title("Plant Weight by Group")
Text(0.5, 1.0, 'Plant Weight by Group')
>>> plt.show()
```

ii.


c. Based on the boxplots in #2b, approximately what percentage of the "trt1" weights are below the minimum "trt2" weight?

i. The minimum trt2 weight is 4.92. Two outlier values that are above 5.5 are clearly visible in the box plot for trt1, and the remaining values look like they are under 4.92. Since we know that there are 10 values in the trt1 group, it looks like approximately 8/10 or 80% of the trt1 weights are below 4.92.

```
>>> min_trt2 = PlantGrowth[PlantGrowth.group == "trt2"].weight.min()
>>> print ("Minimum trt2 Weight: " + str(min_trt2))
Minimum trt2 Weight: 4.92
```

d. Find the exact percentage of the "trt1" weights that are below the minimum "trt2" weight.

i. 80%

ii.
```
>>> trt1_weights = PlantGrowth[PlantGrowth['group'] == 'trt1']['weight']
>>> percentage_below = (trt1_weights < min_trt2).mean() * 100
>>> print ("Percentage of trt1 Weights that Fall Below Minimum trt2 Weight: " + str(percentage_below))
Percentage of trt1 Weights that Fall Below Minimum trt2 Weight: 80.0
```

e. Only including plants with a weight above 5.5, make a barplot of the variable group. Make the barplot colorful using some color palette.

i.

```
>>> heavy_plants = PlantGrowth[PlantGrowth['weight'] > 5.5]
>>> sns.countplot(x='group', data=heavy_plants, palette='Set2')
<stdin>:1: 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.

<Axes: xlabel='group', ylabel='count'>
>>> plt.title('Distribution of Heavy Plants (Weight > 5.5)')
Text(0.5, 1.0, 'Distribution of Heavy Plants (Weight > 5.5)')
>>> plt.show()
```

ii.