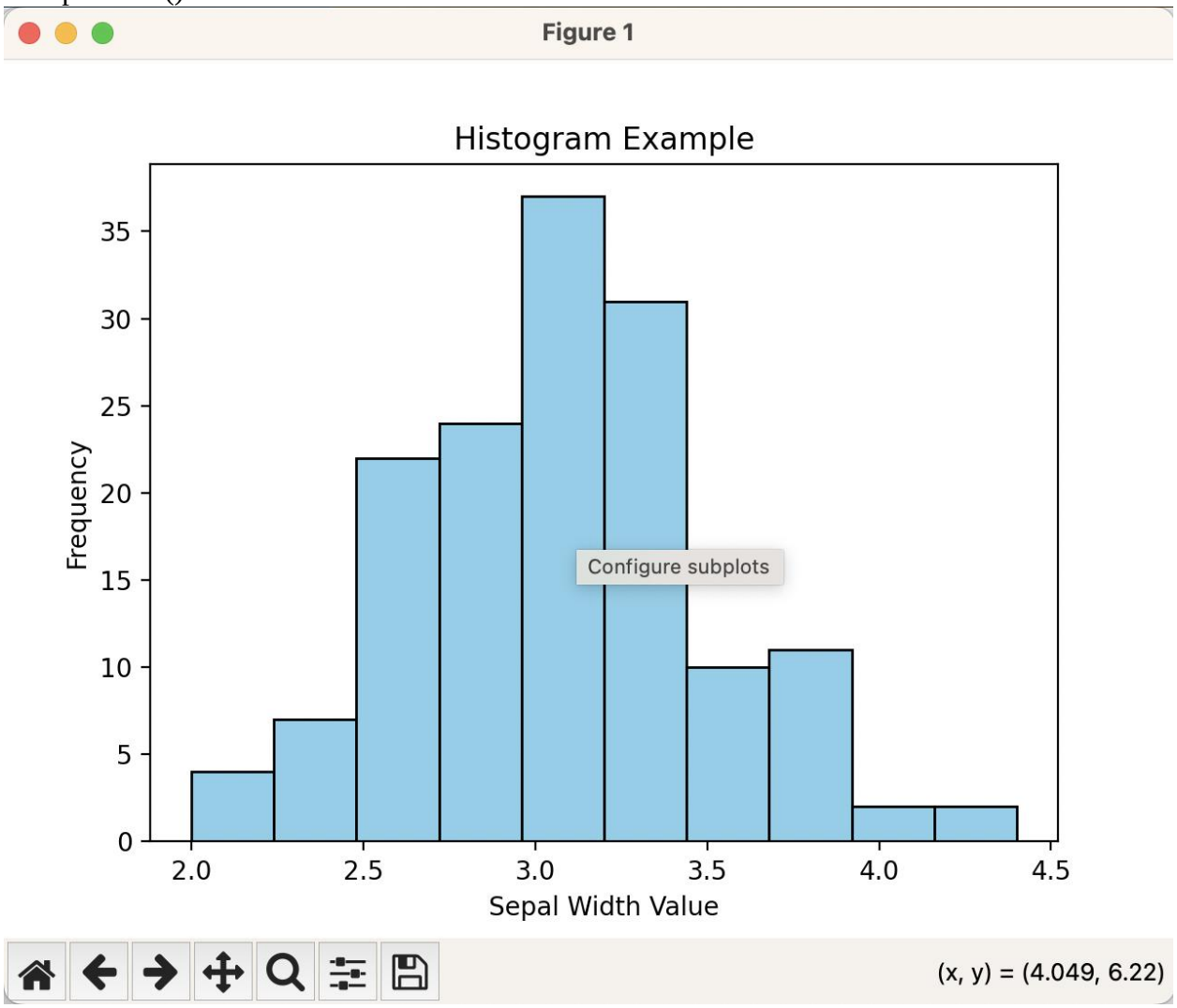
1. Using the `iris` dataset...

   a. Make a histogram of the variable `Sepal.Width`.

I downloaded the iris data and loaded in dataFrame from downloaded csv..

```
>>> import pandas as pd
df = pd.read_csv('iris.csv')
>>> plt.hist(df['sepal.width'], color='skyblue', edgecolor='black')
...
... # Add titles and labels
... plt.title('Histogram Example')
... plt.xlabel('Sepal Width Value')
... plt.ylabel('Frequency')
...
>>> plt.show()
```



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

Ans. Based on the plot, median and mean seems to be the same.

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

Ans.

```
>>> median_sepalWidth = np.median(df['sepal.width'])
... mean_sepalWidth = np.mean(df['sepal.width'])
... print(median_sepalWidth)
... print(mean_sepalWidth)
...
3.0
3.0573333333333337
```

d. Only 27% of the flowers have a `Sepal.Width` higher than _____ cm.

Ans. To find the Sepal.Width value above which only 27% of the flowers lie, we need to calculate the 73rd percentile (100% - 27%) of the Sepal.Width column

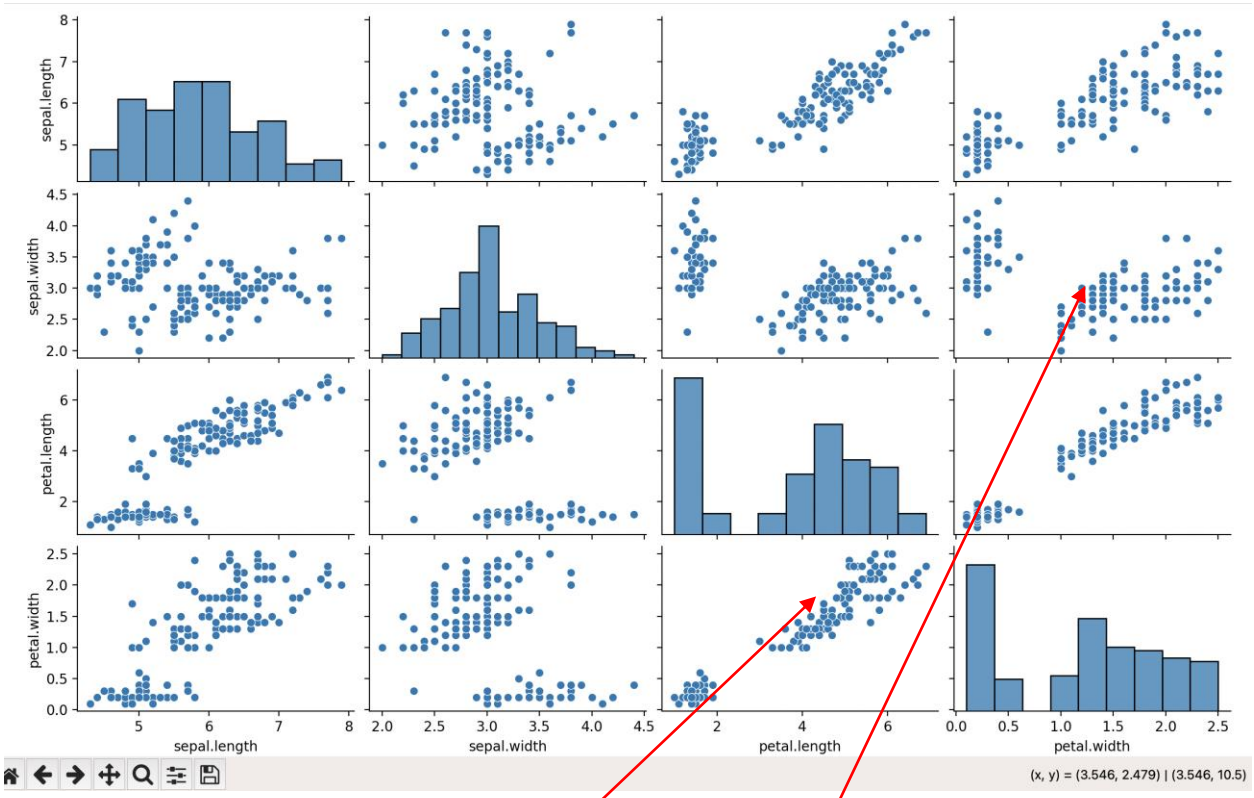
```
>>> sepal_width_73rd_percentile = df['sepal.width'].quantile(.73)
... print(sepal_width_73rd_percentile)
...
3.3
```

27% of flowers have higher than 3.3 cm sepal width

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

```
>>> import seaborn as sns
>>> columns_to_plot = ['sepal.length', 'sepal.width', 'petal.length','petal.width']
... sns.pairplot(df[columns_to_plot], markers='o')
...
<seaborn.axisgrid.PairGrid object at 0x13b593b10>
>>> plt.show()
```

↑



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

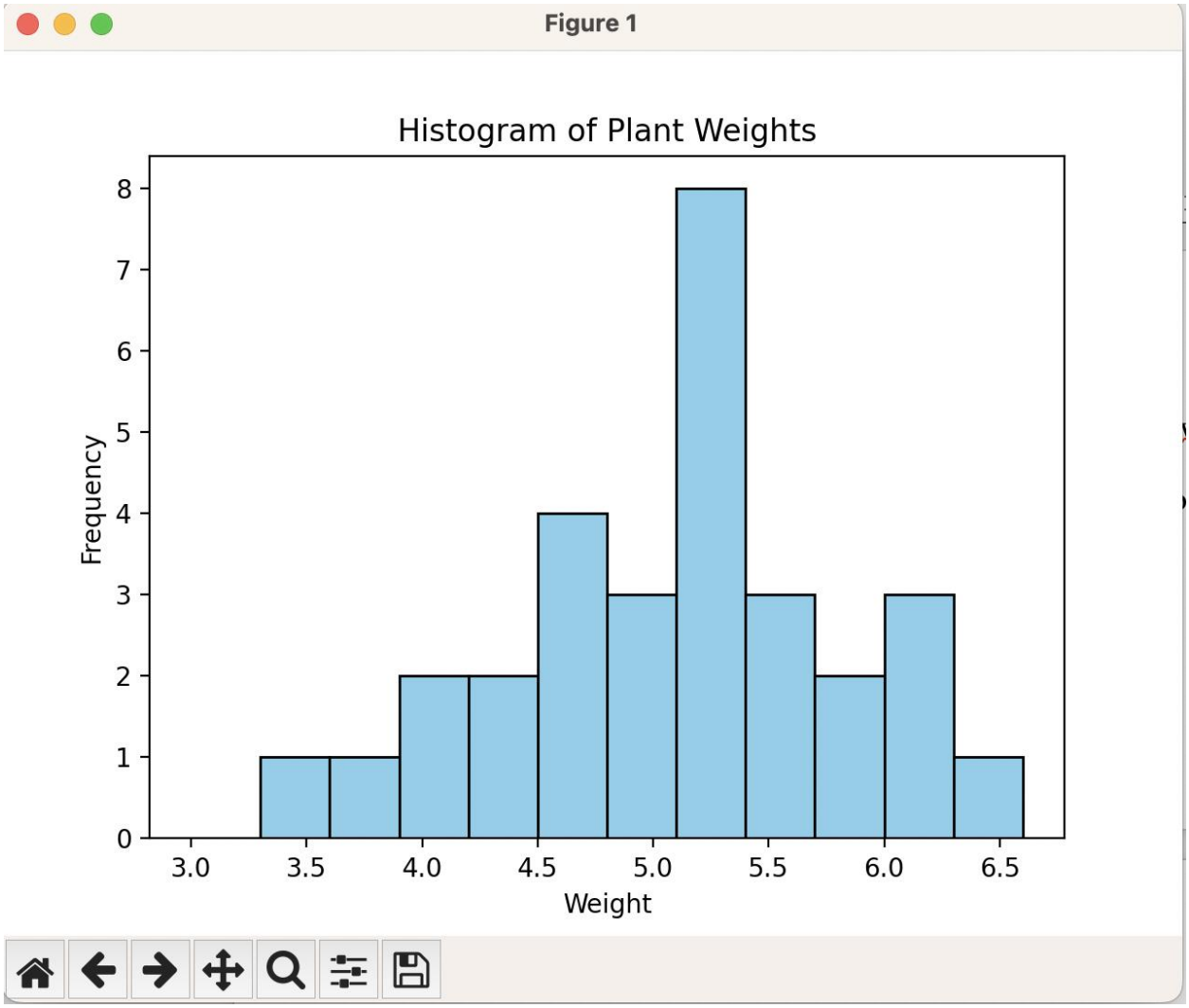
Strong Relationship – petal length and petal width as the relation is linear.
Weak Relationship – sepal length and sepal width. No clear trend of any relation.

2. Using the `PlantGrowth` dataset...

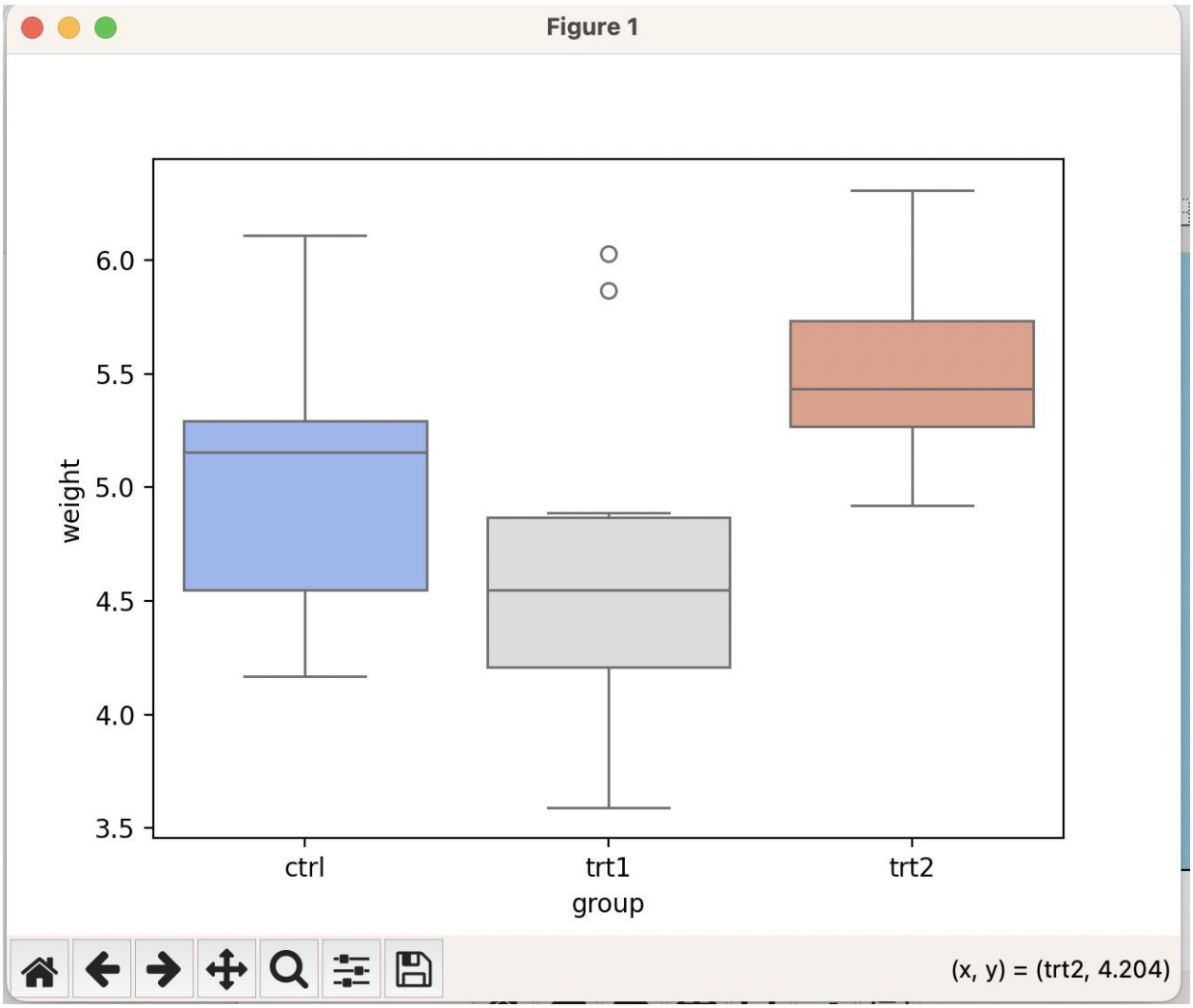
a. Make a histogram of the variable `weight` with breakpoints (bin edges) at every 0.3 units, starting at 3.3.

```
>>> weights=PlantGrowth['weight']
... start_point = 3
... bin_width = .3
... bin_edges = np.arange(start_point,max(weights)+bin_width,bin_width)
...
... plt.hist(weights, bins=bin_edges, edgecolor='black', color='skyblue')
...
... # Add labels and title
... plt.title('Histogram of Plant Weights')
```

... plt.xlabel('Weight')
... plt.ylabel('Frequency')
... plt.show()



b. Make boxplots of `weight` separated by `group` in a single graph.
>>> sns.boxplot(x='group', y='weight', data=PlantGrowth, palette='coolwarm')
... plt.show()



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

Ans. From figure it seems that large percentage of tr1 weights are below minimum of tr2 weights.

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

Ans.

```
>>> #tr1 and tr2 weights
... trt1_weights = PlantGrowth[PlantGrowth['group'] == 'trt1']['weight']
... trt2_weights = PlantGrowth[PlantGrowth['group'] == 'trt2']['weight']
...
... #minimum count
```

```
... min_trt2_weight = trt2_weights.min()
... below_min_count = (trt1_weights < min_trt2_weight).sum()
... print(below_min_count)
... #percentage w.r.t. tr1
... percentage_below = (below_min_count / len(trt1_weights)) * 100
... print(percentage_below)
...
8
80.0
>>>
```
80% of tr1 are below minimum of tr2

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 (in R, try running `?heat.colors` and/or check out https://www.r-bloggers.com/palettes-in-r/).

```
data = PlantGrowth[PlantGrowth['weight'] > 5.5]
... frequency_table = data['group'].value_counts() # create frequency table (equivlanet to table(cyl_cat))
...
... labels_int = frequency_table.index.tolist() # returns the names, which are integers (cylinders) in this case
... labels = list(map(str, labels_int)) # convert those integers to strings
... values = frequency_table.values
... print(frequency_table)
... # Create the bar plot
... sns.barplot(x=labels, y=values, color='red')
... plt.title("Bar Plot (seaborn)")
... plt.xlabel("Group")
... plt.ylabel("Value")
... plt.show()
```

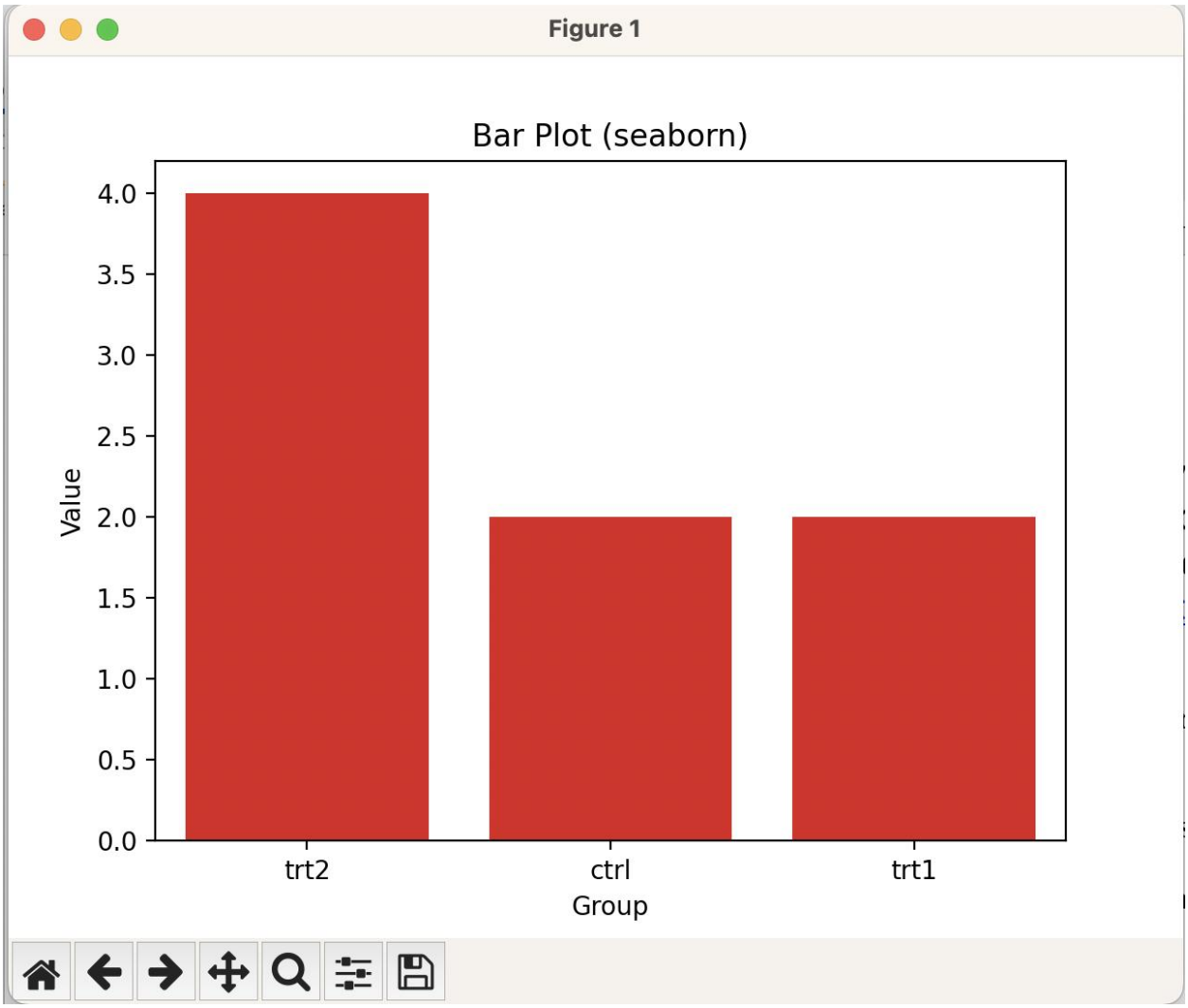


Figure 1
Bar Plot (seaborn)
Value
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
trt2
ctrl
trt1
Group