Contents

1 Plotting practice: histograms

Plot challenge: make a histogram of the results for Test 1 and for Test 2. Draw the average of each test as a straight dashed line.

2 Make a histogram and customize it

1. Open a new DataCamp workspace.
2. Define two lists `test_1` and `test_2`:

   ```
   test_1 = [18.17, 21, 21.5, 21.67, 23.17, 24]
   test_2 = [14.17, 17.67, 17.83, 19.17, 19.5, 23]
   ```

3. Import the library `matplotlib.pyplot` as `plt`:

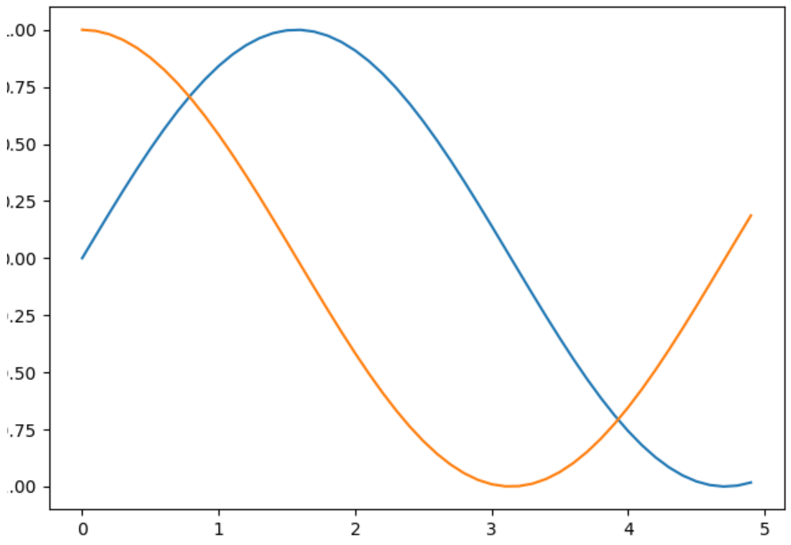
   ```
   import matplotlib.pyplot as plt
   ```

4. Look up the documentation for the `pyplot` module.

   ```
   plt?  # or ?plt
   ```

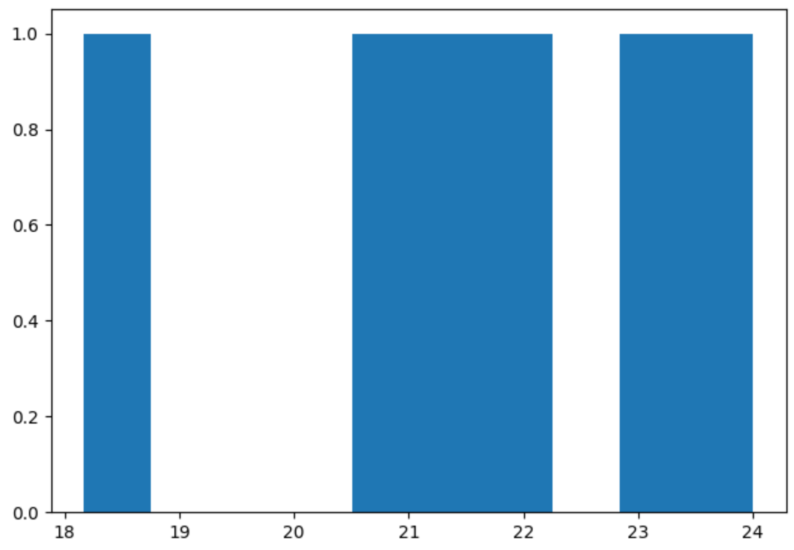
5. Enter and run the first example provided in the help.

   ```
   import numpy as np
   x = np.arange(0,5,0.1)
   y_sin = np.sin(x)
   plt.clf()
   plt.plot(x,y_sin)  # line plot
   y_cos = np.cos(x)
   plt.plot(x,y_cos)
   plt.savefig('../img/plt_demo1.png')
   ```



6. We're after something else: a frequency distribution of a single numeric variable (test points). We use the `plt.hist` method for that - without any bells and whistles at first:

```
plt.clf()
plt.hist(test_1)
plt.savefig('../img/hist_test1.png')
```



7. Add some customization:

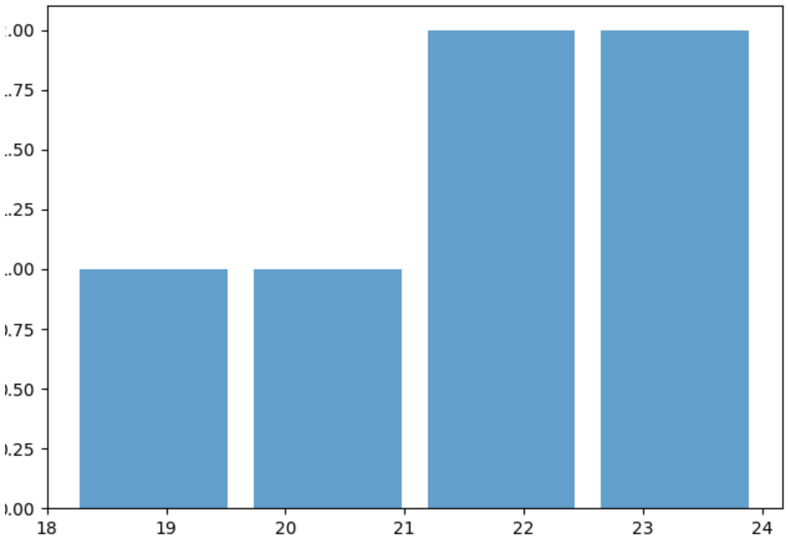
   (a) `plt.hist()` is the function that creates the histogram. The first argument is the data you want to plot.

   (b) The `bins` argument is set to `'auto'` (determine number of bins based on the dataset).

   (c) The `alpha` argument sets the transparency of the bars (1 is opaque, 0 is transparent).

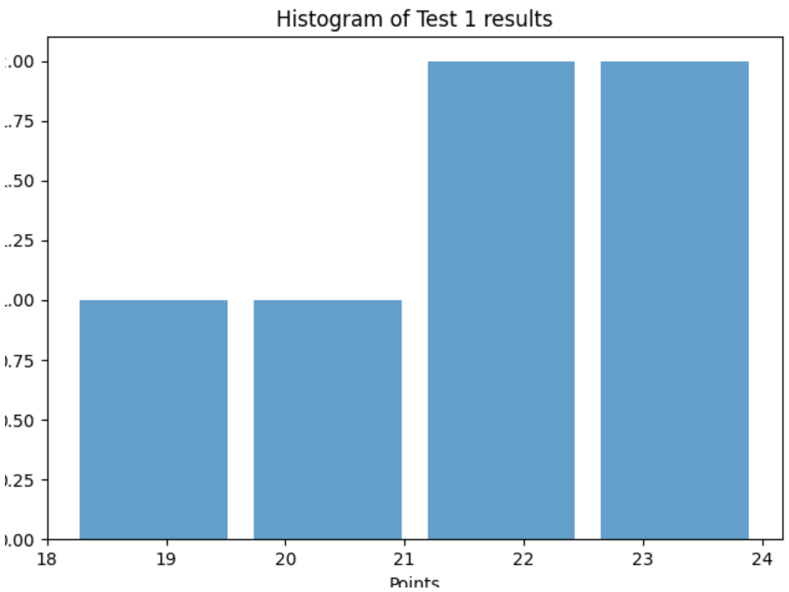
   (d) The `rwidth` argument sets the relative width of the bars as a fraction of the bin width.

```
plt.clf()
plt.hist(test_1, bins='auto', alpha=0.7, rwidth=0.85)
plt.savefig('../img/hist_test_11.png')
```



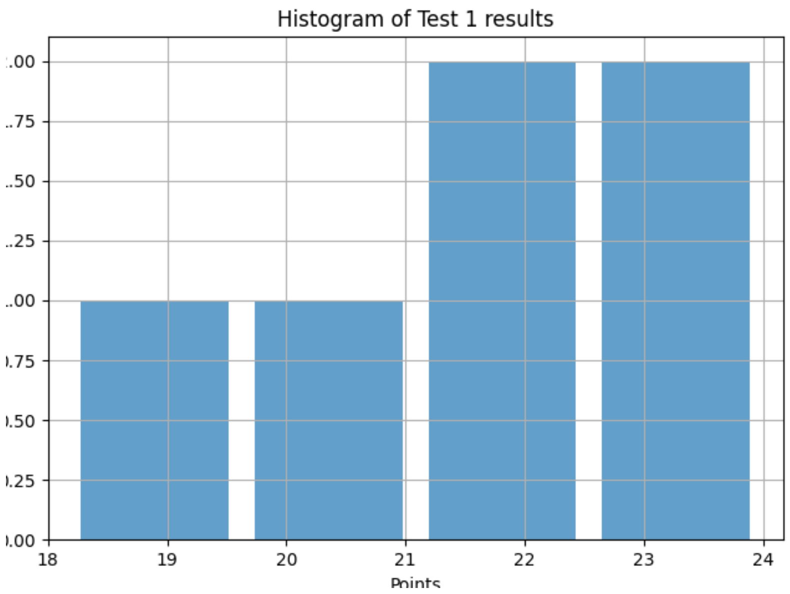
8. Add a title and axis labels:

```
plt.clf()
plt.hist(test_1, bins='auto', alpha=0.7, rwidth=0.85)
plt.xlabel('Points')
plt.ylabel('Frequency')
plt.title('Histogram of Test 1 results')
plt.savefig('../img/hist_test_21.png')
```



9. Finally, put a grid behind the plot to ease readibility:

```
plt.clf()
plt.hist(test_1, bins='auto', alpha=0.7, rwidth=0.85)
plt.xlabel('Points')
plt.ylabel('Frequency')
plt.title('Histogram of Test 1 results')
plt.grid(True)
plt.savefig('../img/hist_test_31.png')
```



10. All of these functions are *methods* of the `pyplot` module.

3 Compute and draw a line for the point average

1. Import the NumPy package as `np`.

   ```
   import numpy as np
   ```

2. Compute the average of the `test_1` and the `test_2` results as `avg_1` and `avg_2` and print them with two digits after the decimal point:

   ```
   avg_1 = np.mean(test_1)
   avg_2 = np.mean(test_2)
   print(f'Average Test 1: {avg_1:.2f}\nAverage Test 2: {avg_2:.2f}')
   ```

   ```
   Average Test 1: 21.58
   Average Test 2: 18.56
   ```

3. Since I already use NumPy, I can do this with an array in one go:

   ```
   data = np.array([test_1, test_2])
   print(data)
   ```

```
[[18.17 21.   21.5  21.67 23.17 24.  ]
 [14.17 17.67 17.83 19.17 19.5  23.  ]]
```

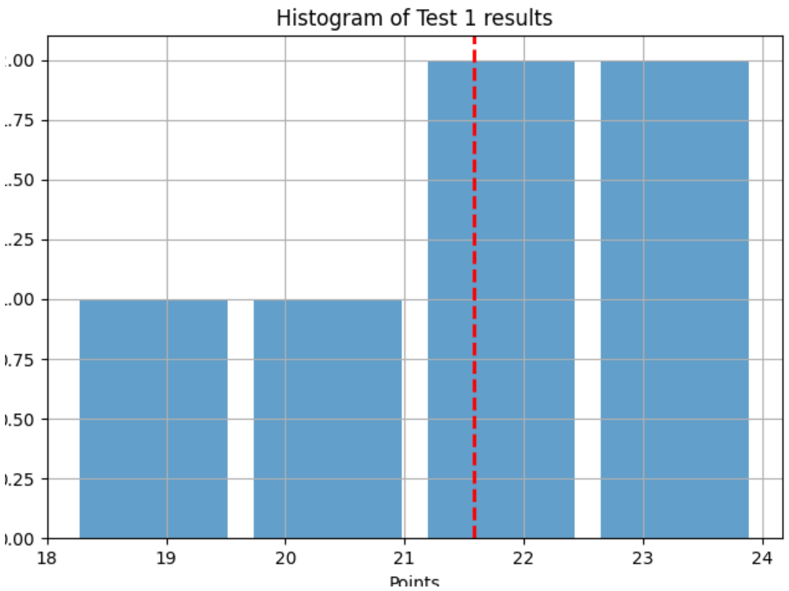
4. To compute the average along the rows, use the `axis=1` parameter:

```
avg = np.mean(data,axis=1)
print(f'Average Test 1: {avg[0]:.2f}\nAverage Test 2: {avg[1]:.2f}')
```

```
Average Test 1: 21.58
Average Test 2: 18.56
```

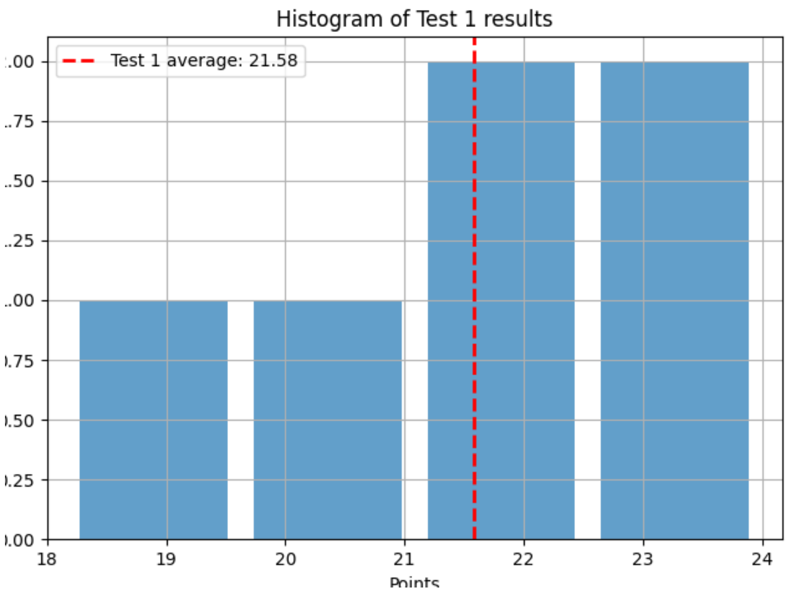
5. We can use `plt.avxline` to print the average as a dashed line into the histogram:

```
plt.clf()
plt.hist(test_1, bins='auto', alpha=0.7, rwidth=0.85)
plt.axvline(avg[0], color='red', linestyle='dashed',linewidth=2)
plt.xlabel('Points')
plt.ylabel('Frequency')
plt.title('Histogram of Test 1 results')
plt.grid(True)
plt.savefig('../img/hist_avg_11.png')
```



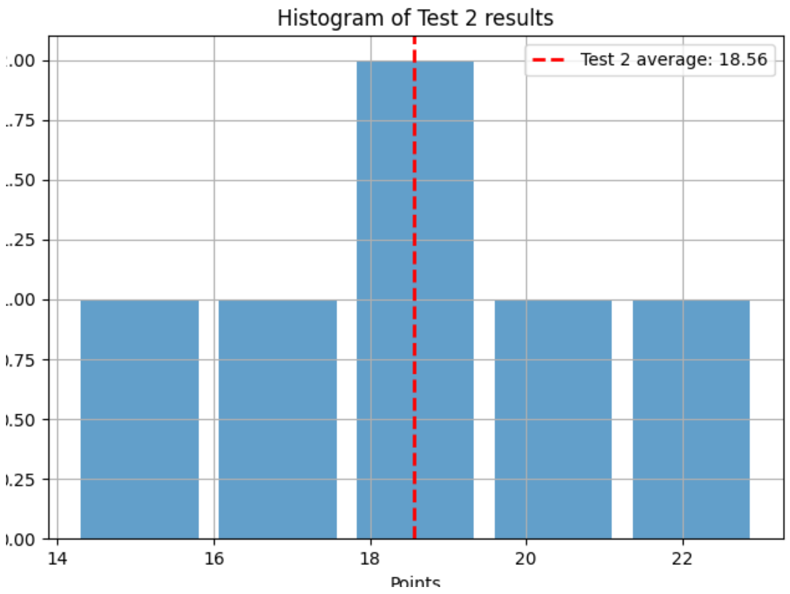
6. Finally, add a legend in the plot itself to identify the average:

```
plt.clf()
plt.hist(test_1, bins='auto', alpha=0.7, rwidth=0.85)
plt.axvline(avg[0], color='r', linestyle='dashed',linewidth=2,
            label=f'Test 1 average: {avg[0]:.2f}')
plt.legend()
plt.xlabel('Points')
plt.ylabel('Frequency')
plt.title('Histogram of Test 1 results')
plt.grid(True)
plt.savefig('../img/hist_avg_21.png')
```



7. Repeat this procedure for the second set of data points and create a similar histogram:

```
plt.clf()
plt.hist(test_2, bins='auto', alpha=0.7, rwidth=0.85)
plt.axvline(avg[1], color='r', linestyle='dashed',linewidth=2,
            label=f'Test 2 average: {avg[1]:.2f}')
plt.legend()
plt.xlabel('Points')
plt.ylabel('Frequency')
plt.title('Histogram of Test 2 results')
plt.grid(True)
plt.savefig('../img/hist_avg_31.png')
```



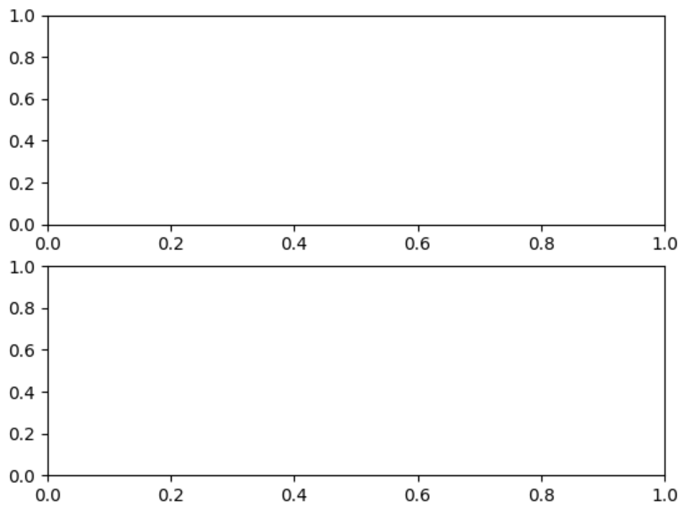
4 Subplots

We want to put the two histogram plots next to one another on two panels. To do this, we need to use the `plt.subplots` method.

1. Create a figure and a set of two subplots (for each dataset):

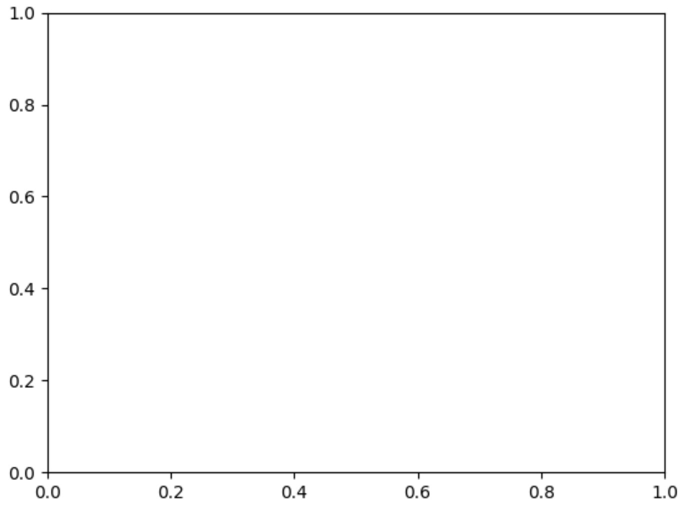
    - `plt.subplots(2)` creates a new figure `fig` and returns a NumPy array `axs` containing the created subplot objects.
    - `fig` is the whole window or page that everything is drawn on.
    - `axs` is an array of length 2 containing the axes for the subplots. In this case, since you're creating 2 subplots, `axs` will be an array of length 2. Each item in the array is a separate set of axes, which you can think of as an individual plot. You can draw on these axes (i.e., create a plot) by calling methods on them.

```
plt.clf()
fig, axs = plt.subplots(2)
plt.savefig('../img/subplot2.png')
```



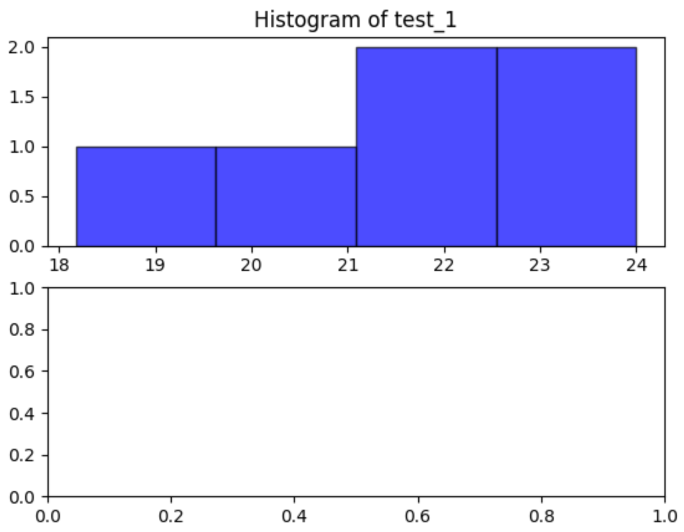
2. When you run the code you should see two empty plot panels. You can experiment with these to find out more about `plt` after looking at `plt.subplots?`. If you remove the `subplot` argument, you get one, not two plots.

   ```
   plt.clf()
   fig, axs = plt.subplots()
   plt.savefig('../img/subplot1.png')
   ```



3. To plot a histogram of your data on a subplot N of your figure, you call `axs[N].hist()`. Do this now for N=0 only:

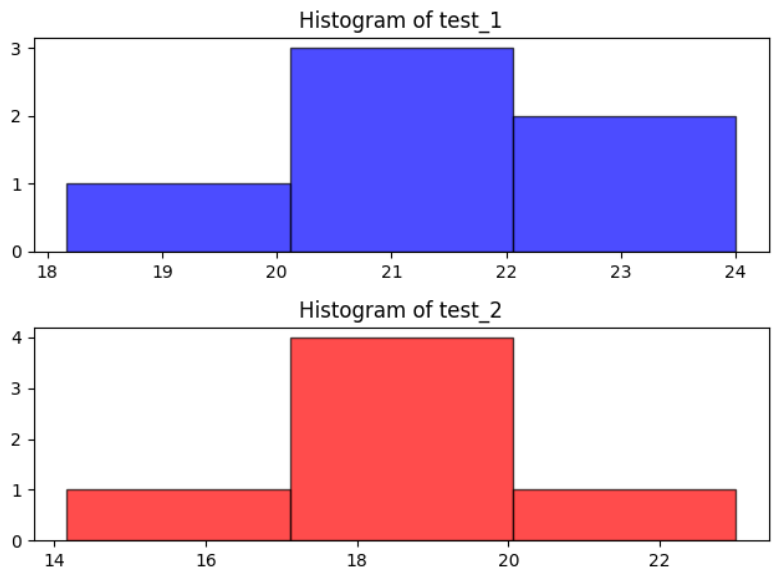
```
plt.clf()
# Create a figure and a set of subplots
fig, axs = plt.subplots(2)

# Create a histogram for test_1
axs[0].hist(test_1,
            bins='auto',
            color='b',
            alpha=0.7,
            edgecolor='black')
axs[0].set_title('Histogram of test_1')
plt.savefig('../img/sub_hist_11.png')
```



4. Now add the code for the second histogram below it, adapting the values accordingly:

```
import matplotlib.pyplot as plt
import numpy as np

# input data as lists
test_1 = [18.17, 21, 21.5, 21.67, 23.17, 24]
test_2 = [14.17, 17.67, 17.83, 19.17, 19.5, 23]

plt.clf()
# Create a figure and a set of subplots
fig, axs = plt.subplots(2)

# Create a histogram for test_1
axs[0].hist(test_1,
            bins=3,
            color='b',
            alpha=0.7,
            edgecolor='black')
axs[0].set_title('Histogram of test_1')
```

```
# Create a histogram for test_2
axs[1].hist(test_2, bins=3, color='r', alpha=0.7, edgecolor='black')
axs[1].set_title('Histogram of test_2')

# Display the (tight) plot
plt.tight_layout()
plt.savefig('../img/sub_hist_21.png')
```

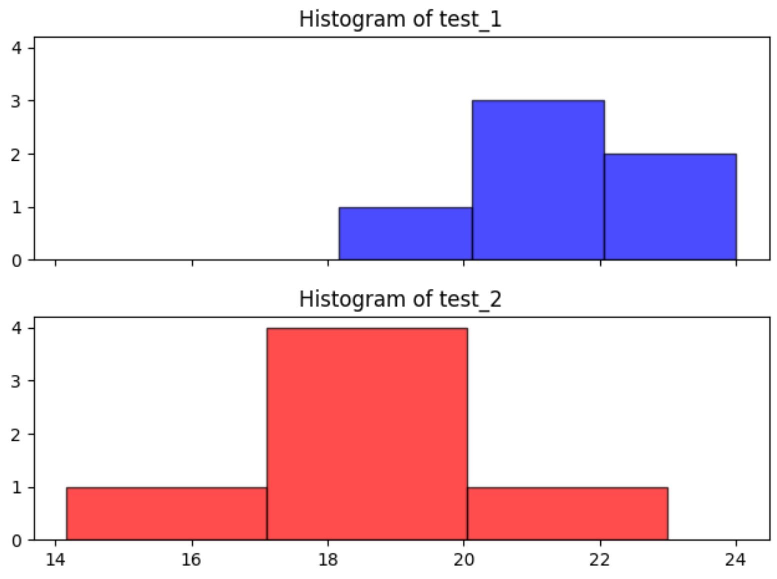


5. Here, `plt.tight_layout()` automatically adjusts subplot parameters so that the subplot fits the panels nicely. Take it out and re-plot to see the effect.

6. We've still got a problem: it is not easy to compare the two histograms because both x and y scales are different. To align the x and y scales of the two plots, you can use the `sharex` and `sharey` parameters when creating the subplots:

```
import matplotlib.pyplot as plt
import numpy as np

# input data as lists
test_1 = [18.17, 21, 21.5, 21.67, 23.17, 24]
```

```
test_2 = [14.17, 17.67, 17.83, 19.17, 19.5, 23]

plt.clf()
# Create a figure and a set of subplots
fig, axs = plt.subplots(2, sharex=True, sharey=True)

# Create a histogram for test_1
axs[0].hist(test_1,
            bins=3,
            color='b',
            alpha=0.7,
            edgecolor='black')
axs[0].set_title('Histogram of test_1')

# Create a histogram for test_2
axs[1].hist(test_2, bins=3, color='r', alpha=0.7, edgecolor='black')
axs[1].set_title('Histogram of test_2')

# Display the (tight) plot
plt.tight_layout()
plt.savefig('../img/sub_hist_31.png')
```


Histogram of test_1
Histogram of test_2