6nxwocv1n

February 16, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Find arithmetic mean of 20, 2, 7, 1, 34

[2]:
```
arr = np.array([20,2,7,1,34])
mean = np.mean(arr)
print(mean)
```

```
12.8
```

2. Create following matrix using multidimensional array and calculate arithmetic mean for each column, each row and considering entire data.

- 14 17 12 33 44
- 15 6 27 8 19
- 23 2 54 1 4

[3]:
```
arr = np.array([[14,17,12,33,44], [15,6,27,8,19],[23,2,54,1,4]])
r1 =np.mean(arr[0])
r2 = np.mean(arr[1])
r3 = np.mean(arr[2])
c1 = np.mean(arr[:,0])
c2 = np.mean(arr[:,1])
c3 = np.mean(arr[:,2])
c4 = np.mean(arr[:,3])
c5 = np.mean(arr[:,4])
print("The row means are: ",r1,", ",r2," and ",r3)
print("The columns means are: ",c1,", ",c2,", ",c3,", ",c4,", ",c5)
```

```
The row means are:  24.0 ,  15.0  and  16.8
The columns means are:  17.333333333333332 ,  8.333333333333334 ,  31.0 ,  14.0
,  22.333333333333332
```

3. Find minimum value, maximum value and range for entire data, column wise and row wise.

- 3,7,5
- 8,4,3
- 2,4,9

[4]:
```
arr2 = np.array([[3,7,5], [8,4,3],[2,4,9]])
min = np.min(arr2)
max = np.max(arr2)
range =(min, max)
minc1 = np.mean(arr2[:,0])
minc2 =np.mean(arr2[:,1])
minc3 = np.mean(arr2[:,2])
print("The min, max, range and min columns for columns 1,2 and 3 are:␣
 ↪",min,max,range,minc1,minc2,minc3)
```

```
The min, max, range and min columns for columns 1,2 and 3 are:  2 9 (2, 9)
4.333333333333333 5.0 5.666666666666667
```

4. Find weighted average for the data given below.

Outcomes Frequency 1 4 2 3 3 2 4 1

[5]:
```
out = np.array([1,2,3,4])
freq = np.array([4,3,2,1])

x = out * freq

print("The mean is: " , sum(x)/ sum(freq))
```

```
The mean is:  2.0
```

5. The speed of 13 vehicles is 99,86,87,88,111,86,103,87,94,78,77,85,86 Find mean, median and mode.

[6]:
```
from scipy import stats
speeds = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
print("Mean: ", np.mean(speeds))
print("Median: ", np.median(speeds))
print("Mode: ", stats.mode(speeds)[0])
```

```
Mean:  89.76923076923077
Median:  87.0
Mode:  86
```

6. Calculate geometric mean for each column, each row and considering entire data.

- 1 3 27
- 3 4 6
- 7 6 3
- 3 6 8

[7]:
```
data = np.array([[1,3,27],
                 [3,4,6],
                 [7,6,3],
                 [3,6,8]])
print("The Geometric mean of columns is: ", stats.gmean(data,axis= 0))
```

```
print("The Geometric mean of rows is: ", stats.gmean(data,axis= 1))
print("The Geometric mean is: ", stats.gmean(data.flatten()))
```

```
The Geometric mean of columns is:  [2.81731325 4.55901411 7.89644408]
The Geometric mean of rows is:  [4.32674871 4.16016765 5.01329793 5.24148279]
The Geometric mean is:  4.66350606643965
```

7. Calculate harmonic mean for 1, 3, 5, 7, 9

[8]:
```
hm = np.array([1,3,5,7,9])
print("The harmonic mean is: ", stats.hmean(hm))
```

```
The harmonic mean is:  2.7975133214920076
```

8. Calculate median for each column, each row and considering entire data.

- 30 65 70
- 80 95 10
- 50 90 60

[9]:
```
data = np.array([[30,65,70],
                 [80,95,10],
                 [50,90,60]])

r1 =np.median(data[0])
r2 = np.median(data[1])
r3 = np.median(data[2])
c1 = np.median(data[:,0])
c2 = np.median(data[:,1])
c3 = np.median(data[:,2])
print("The row medians are: ",r1,", ",r2," and ",r3)
print("The columns medians are: ",c1,", ",c2,", ",c3)
```

```
The row medians are:  65.0 ,  80.0  and  60.0
The columns medians are:  50.0 ,  90.0 ,  60.0
```

[10]:
```
from IPython.display import Image
Image(filename='q9.png')
```

[10]:

9. The number of solar heating systems available to the public is quite large, and their heat-storage capacities are quite varied. Here is a distribution of heat-storage capacity (in days) of 28 systems that were tested recently by University Laboratories, Inc.:

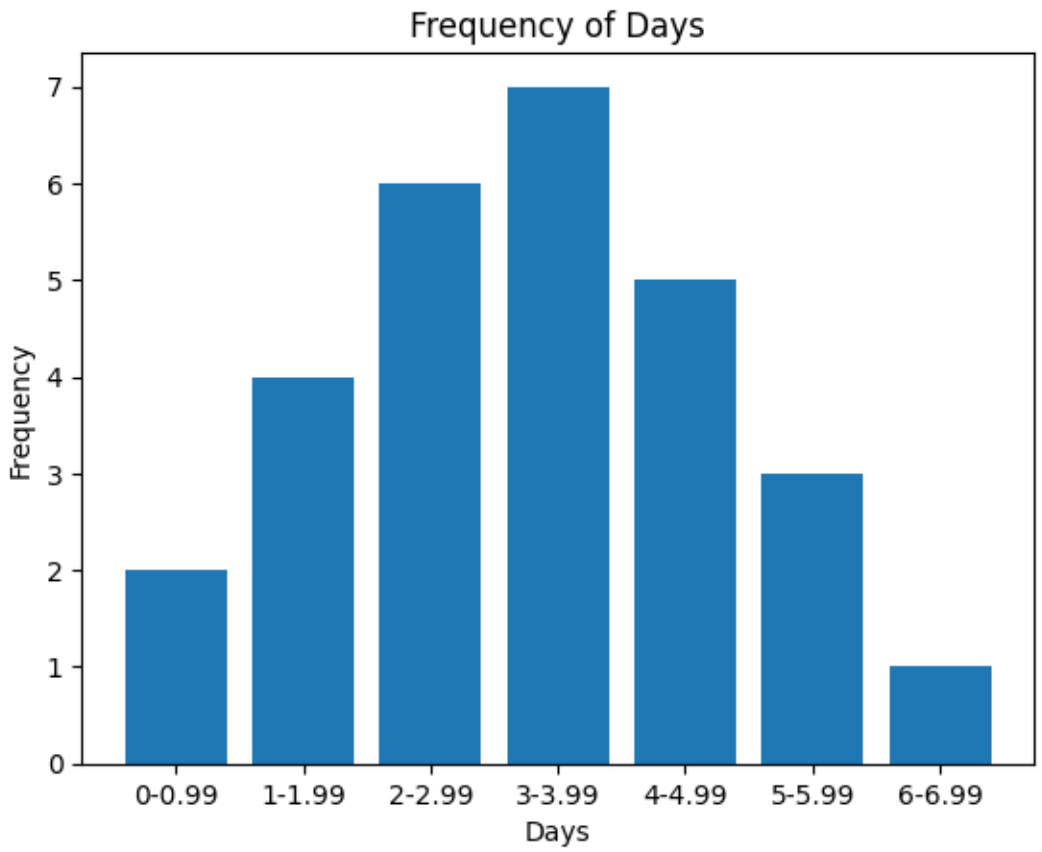
Days	Frequency
0–0.99	2
1–1.99	4
2–2.99	6
3–3.99	7
4–4.99	5
5–5.99	3
6–6.99	1

University Laboratories, Inc., knows that its report on the tests will be widely circulated and used as the basis for tax legislation on solar-heat allowances. It therefore wants the measures it uses to be as reflective of the data as possible.
(a) Compute the mean for these data.
(b) Compute the mode for these data.
(c) Compute the median for these data.
(d) Select the answer among parts (a), (b), and (c) that best reflects the central tendency of the test data and justify your choice.

[11]:
```
# Given frequency distribution
data = [
    (0.00, 0.99, 2),
    (1.00, 1.99, 4),
    (2.00, 2.99, 6),
    (3.00, 3.99, 7),
    (4.00, 4.99, 5),
    (5.00, 5.99, 3),
    (6.00, 6.99, 1)
]

expanded_data = []
for lower, upper, freq in data:
    midpoint = (lower + upper) / 2
    expanded_data.extend([midpoint] * freq)

mean_value = np.mean(expanded_data)
```

```
mode_value = stats.mode(expanded_data, keepdims=True).mode[0]

median_value = np.median(expanded_data)

# Print results
print(f"Mean: {mean_value:.2f}")
print(f"Mode: {mode_value}")
print(f"Median: {median_value:.2f}")

days = ['0-0.99', '1-1.99', '2-2.99', '3-3.99', '4-4.99', '5-5.99', '6-6.99']
frequency = [2, 4, 6, 7, 5, 3, 1]
# Create the bar chart
plt.bar(days, frequency)

# Set the title and labels
plt.title('Frequency of Days')
plt.xlabel('Days')
plt.ylabel('Frequency')

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

```
Mean: 3.28
Mode: 3.495
Median: 3.50
```



10. Write a function for calculating percentile and determine 30, 50, 75 and 90 percentiles for the following data. 30,40,72,83,25,10,50,90,60,15,5,9,34,23,67,80,67,45

[12]:
```
def calculate_percentiles(data, percentiles):
    return np.percentile(data, percentiles)

data = [30, 40, 72, 83, 25, 10, 50, 90, 60, 15, 5, 9, 34, 23, 67, 80, 67, 45]
percentiles = [30, 50, 75, 90]

percentile_values = calculate_percentiles(data, percentiles)
print(f"30th percentile: {percentile_values[0]}")
print(f"50th percentile: {percentile_values[1]}")
print(f"75th percentile: {percentile_values[2]}")
print(f"90th percentile: {percentile_values[3]}")
```

```
30th percentile: 25.5
50th percentile: 42.5
75th percentile: 67.0
90th percentile: 80.9
```

11. The numbers of apartments in 27 apartment complexes in Cary, North Carolina, are given

below. 91 79 66 98 127 139 154 147 192 88 97 92 87 142 127 184 145 162 95 89 86 98 145 129 149 158 241 (a) Construct a frequency distribution using intervals 66–87, 88–109, 220–241. (b) Estimate the modal value (c) Compute the mean of the raw data. (d) Compare your, answers in parts (b) and (c) and comment on which of the two is the better measure of central tendency of these data and why.

[13]:
```
apartments = [91, 79, 66, 98, 127, 139, 154, 147, 192, 88, 97, 92, 87, 142,␣
 ↪127, 184, 145, 162, 95, 89, 86, 98, 145, 129, 149, 158, 241]

# (a) Construct a frequency distribution using intervals 66-87, 88-109, 220-241
intervals = [(66, 87), (88, 109), (220, 241)]
frequency_distribution = {f"{interval[0]}-{interval[1]}": 0 for interval in␣
 ↪intervals}

for apartment in apartments:
    for interval in intervals:
        if interval[0] <= apartment <= interval[1]:
            frequency_distribution[f"{interval[0]}-{interval[1]}"] += 1

print("Frequency Distribution:", frequency_distribution)

# (b) Estimate the modal value
mode_value = stats.mode(apartments, keepdims=True).mode[0]
print("Estimated Modal Value:", mode_value)

# (c) Compute the mean of the raw data
mean_value = np.mean(apartments)
print("Mean of the Raw Data:", mean_value)

# (d) Compare the answers in parts (b) and (c) and comment on which of the two␣
 ↪is the better measure of central tendency of these data and why.
print("The mode represents the most frequently occurring value, while the mean␣
 ↪represents the average value of the data set. Depending on the distribution␣
 ↪of the data, one may be more representative than the other. In this case,␣
 ↪the mean provides a better measure of central tendency as it takes into␣
 ↪account all values in the data set.")
```

```
Frequency Distribution: {'66-87': 4, '88-109': 8, '220-241': 1}
Estimated Modal Value: 98
Mean of the Raw Data: 126.18518518518519
The mode represents the most frequently occurring value, while the mean
represents the average value of the data set. Depending on the distribution of
the data, one may be more representative than the other. In this case, the mean
provides a better measure of central tendency as it takes into account all
values in the data set.
```

12. There are 39 plants in the garden. A few plants were selected randomly and their heights in cm were recorded as follows: 51, 38, 79, 46, 57. Calculate the standard deviation of their

heights.

[14]:
```
flo = np.array([51, 38, 79, 46, 57])
print("The standard deviation of the heights is : " , np.std(flo))
```

```
The standard deviation of the heights is :  13.876599006961325
```

13. The Casual Life Insurance Company is considering purchasing a new fleet of company cars. The financial department's director, Tom Dawkins, sampled 40 employees to determine the number of miles each drove over a 1-year period. The results of the study follow. Calculate the range and interquartile range

[15]:
```
mileage_data = [
    3600, 4200, 4700, 4900, 5300, 5700, 6700, 7300,
    7700, 8100, 8300, 8400, 8700, 8700, 8900, 9300,
    9500, 9500, 9700, 10000, 10300, 10500, 10700, 10800,
    11000, 11300, 11300, 11800, 12100, 12700, 12900, 13100,
    13500, 13800, 14600, 14900, 16300, 17200, 18500, 20300
]

range_value = np.max(mileage_data) - np.min(mileage_data)  # Manual calculation␣
 ↪of range

q1 = np.percentile(mileage_data, 25)
q3 = np.percentile(mileage_data, 75)
iqr_range = (q1, q3)


print(f"Range: {range_value}")
print(f"Interquartile Range (IQR): {iqr_range}")
```

```
Range: 16700
Interquartile Range (IQR): (8250.0, 12750.0)
```

14. The head chef of The Flying Taco has just received two dozen tomatoes from her supplier, but she isn't ready to accept them. She knows from the invoice that the average weight of a tomato is 7.5 ounces, but she insists that all be of uniform weight. She will accept them only if the average weight is 7.5 ounces and the standard deviation is less than 0.5 ounce. Here are the weights of the tomatoes 6.3 7.2 7.3 8.1 7.8 6.8 7.5 7.8 7.2 7.5 8.1 8.2 8.0 7.4 7.6 7.7 7.6 7.4 7.5 8.4 7.4 7.6 6.2 7.4 What is the chef's decision and why?

[16]:
```
tomato_weights = [
    6.3, 7.2, 7.3, 8.1, 7.8, 6.8, 7.5, 7.8, 7.2, 7.5, 8.1, 8.2,
    8.0, 7.4, 7.6, 7.7, 7.6, 7.4, 7.5, 8.4, 7.4, 7.6, 6.2, 7.4
]
mean_weight = np.mean(tomato_weights)
std_dev = np.std(tomato_weights)

acceptable_mean = 7.5
```

```
acceptable_std_dev = 0.5

if mean_weight == acceptable_mean and std_dev < acceptable_std_dev:
    decision = "Accept the tomatoes."
else:
    decision = "Reject the tomatoes."

print(f"Mean Weight: {mean_weight:.2f} ounces")
print(f"Standard Deviation: {std_dev:.2f} ounces")
print(f"Decision: {decision}")
```

```
Mean Weight: 7.50 ounces
Standard Deviation: 0.52 ounces
Decision: Reject the tomatoes.
```

15. A company is considering employing one of two training programs. Two groups were trained for the same task. Group 1 was trained by program A; group 2, by program B. For the first group, the times required to train the employees had an average of 32.11 hours and a variance of 68.09. In the second group, the average was 19.75 hours and the variance was 71.14. Which training program has less relative variability in its performance?

[17]:
```
import math

mean_program_A = 32.11
variance_program_A = 68.09
std_dev_program_A = math.sqrt(variance_program_A)

mean_program_B = 19.75
variance_program_B = 71.14
std_dev_program_B = math.sqrt(variance_program_B)

cv_program_A = (std_dev_program_A / mean_program_A) * 100
cv_program_B = (std_dev_program_B / mean_program_B) * 100

if cv_program_A < cv_program_B:
    decision = "Program A has less relative variability."
else:
    decision = "Program B has less relative variability."

print(f"Coefficient of Variation for Program A: {cv_program_A:.2f}%")
print(f"Coefficient of Variation for Program B: {cv_program_B:.2f}%")
print(f"Decision: {decision}")
```

```
Coefficient of Variation for Program A: 25.70%
Coefficient of Variation for Program B: 42.71%
Decision: Program A has less relative variability.
```

[18]:
```
Image(filename='q16.png')
```

[18]:

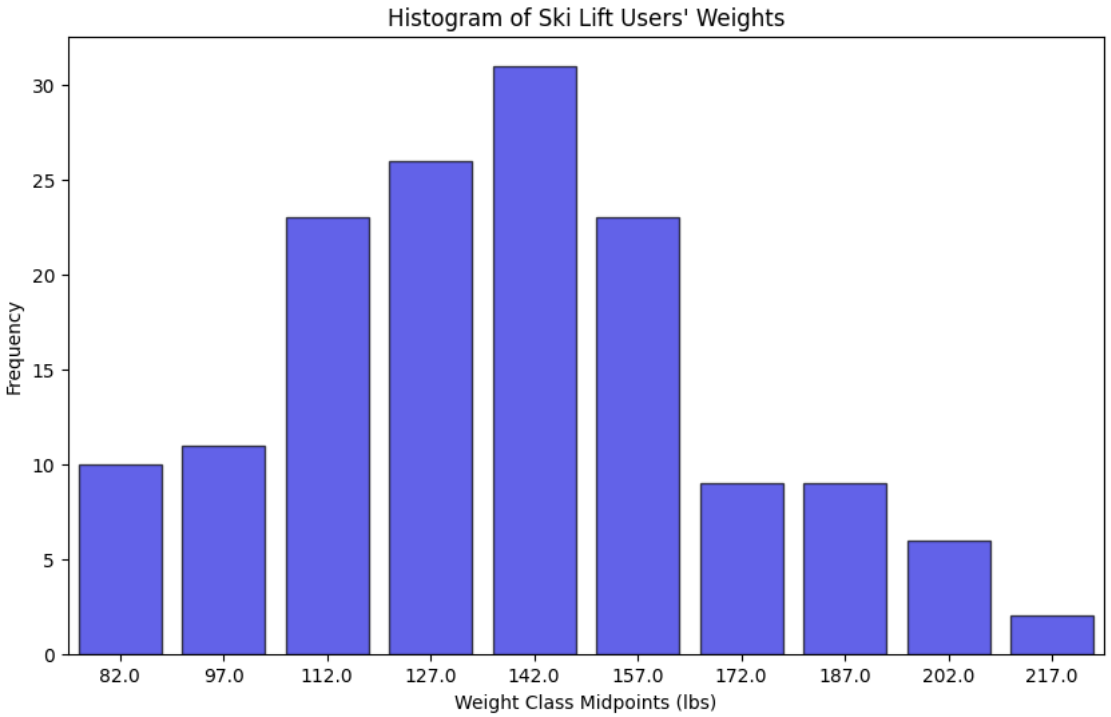
16. Here is a frequency distribution of the weight of 150 people who used a ski lift a certain day. Construct a histogram for these data.

Class	Frequency	Class	Frequency
75–89	10	150–164	23
90–104	11	165–179	9
105–119	23	180–194	9
120–134	26	195–209	6
135–149	31	210–224	2

(a) What can you see from the histogram about the data that was not immediately apparent from the frequency distribution?
(b) If each ski lift chair holds two people but is limited in total safe weight capacity to 400 pounds, what can the operator do to maximize the people capacity of the ski lift without exceeding the safe weight capacity of a chair? Do the data support your proposal?

[19]:

```
weight_classes = [(75, 89), (90, 104), (105, 119), (120, 134), (135, 149),
                  (150, 164), (165, 179), (180, 194), (195, 209), (210, 224)]
frequencies = [10, 11, 23, 26, 31, 23, 9, 9, 6, 2]

midpoints = [(low + high) / 2 for low, high in weight_classes]

plt.figure(figsize=(10, 6))
sns.barplot(x=midpoints, y=frequencies, color='blue', edgecolor='black',␣
 ↪alpha=0.7)

plt.xlabel("Weight Class Midpoints (lbs)")
plt.ylabel("Frequency")
plt.title("Histogram of Ski Lift Users' Weights")
plt.show()
```



17. Test scores for a college statistics class held during the day are: 99 56 78 55.5 32 90 80 81 56 59 45 77 84.5 84 70 72 68 32 79 90 Test scores for a college statistics class held during the evening are: 98 78 68 83 81 89 88 76 65 45 98 90 80 84.5 85 79 78 98 90 79 81 25.5

- a. Find the smallest and largest values, the median, and the first and third quartile for the day class.
- b. Find the smallest and largest values, the median, and the first and third quartile for the night class.
- c. For each data set, what percentage of the data is between the smallest value and the first quartile? the first quartile and the median? the median and the third quartile? the third quartile and the largest value? What percentage of the data is between the first quartile and the largest value?
- d. Create a box plot for each set of data. Use one number line for both box plots.
- e. Which box plot has the widest spread for the middle 50% of the data (the data between the first and third quartiles)? What does this mean for that set of data in comparison to the other set of data.

[20]:
```
scores_day = [99, 56, 78, 55.5, 32, 90, 80, 81, 56, 59, 45, 77, 84.5, 84, 70,␣
 ↪72, 68, 32, 79, 90]
scores_night = [98, 78, 68, 83, 81, 89, 88, 76, 65, 45, 98, 90, 80, 84.5, 85,␣
 ↪79, 78, 98, 90, 79, 81, 25.5]

def summary_statistics(data):
```

```
    min_val = np.min(data)
    max_val = np.max(data)
    q1 = np.percentile(data, 25)
    median = np.median(data)
    q3 = np.percentile(data, 75)
    return min_val, q1, median, q3, max_val

# Compute statistics for both groups
stats_day = summary_statistics(scores_day)
stats_night = summary_statistics(scores_night)

# Compute percentage ranges
def percentage_ranges(data):
    q1 = np.percentile(data, 25)
    median = np.median(data)
    q3 = np.percentile(data, 75)
    min_val = np.min(data)
    max_val = np.max(data)
    total = len(data)
    return {
        "Min to Q1": np.sum((data >= min_val) & (data < q1)) / total * 100,
        "Q1 to Median": np.sum((data >= q1) & (data < median)) / total * 100,
        "Median to Q3": np.sum((data >= median) & (data < q3)) / total * 100,
        "Q3 to Max": np.sum((data >= q3) & (data <= max_val)) / total * 100,
        "Q1 to Max": np.sum((data >= q1) & (data <= max_val)) / total * 100
    }

percentages_day = percentage_ranges(scores_day)
percentages_night = percentage_ranges(scores_night)



iqr_day = stats_day[3] - stats_day[1]
iqr_night = stats_night[3] - stats_night[1]
spread_comparison = "Day class has a wider spread." if iqr_day > iqr_night else␣
 ↪"Night class has a wider spread."

print("Summary Statistics (Day Class):", stats_day)
print("Summary Statistics (Night Class):", stats_night)
print("Percentage Ranges (Day Class):", percentages_day)
print("Percentage Ranges (Night Class):", percentages_night)
print("Middle 50% Spread Comparison:", spread_comparison)

plt.figure(figsize=(10, 6))
sns.boxplot(data=[scores_day, scores_night], palette=["blue", "orange"])
plt.xticks([0, 1], ["Day Class", "Night Class"])
plt.xlabel("Class")
```

```
plt.ylabel("Scores")
plt.title("Box Plots of Test Scores for Day and Night Classes")
plt.show()
```

```
Summary Statistics (Day Class): (32.0, 56.0, 74.5, 81.75, 99.0)
Summary Statistics (Night Class): (25.5, 78.0, 81.0, 88.75, 98.0)
Percentage Ranges (Day Class): {'Min to Q1': 20.0, 'Q1 to Median': 30.0, 'Median
to Q3': 25.0, 'Q3 to Max': 25.0, 'Q1 to Max': 80.0}
Percentage Ranges (Night Class): {'Min to Q1': 22.727272727272727, 'Q1 to
Median': 22.727272727272727, 'Median to Q3': 27.27272727272727, 'Q3 to Max':
27.27272727272727, 'Q1 to Max': 77.27272727272727}
Middle 50% Spread Comparison: Day class has a wider spread.
```