centralValueLocation

November 14, 2024

Central Value Location: - It refers to the methods used to find a central or typical value that represents the center of a data set, giving an overview of where most values are concentrated. It's an essential concept in descriptive statistics, helping us summarize a large data set with a single value that best represents the entire set.

There are several measures for locating central values, each suited to different types of data or distributions:

Mean - The arithmetic average, calculated by summing all values and dividing by the number of values. It is sensitive to extreme values or outliers.

Median - The middle value when data is ordered from smallest to largest. The median is robust to outliers, making it useful for skewed distributions.

Mode - The most frequently occurring value in a data set. The mode is useful for categorical data or finding the most common value in a set.

Trimmed Mean - A variation of the mean that excludes a certain percentage of the highest and lowest values to reduce the impact of outliers.

Weighted Average - An average where each data point contributes differently based on a weight assigned to it, useful when some values are more important than others.

Each of these methods provides different insights into the data's central tendency, and choosing the right measure depends on the data's nature and the analysis goals.

1. Mean
    - Definition: The mean is the arithmetic average of a set of values. It's one of the most common measures of central tendency.
    - Formula:
        - Mean = (Σ x_i) / n
        - Where:
            - x_i = each data point
            - n = number of data points
    - Example:
        - For the data set: [4, 7, 10, 5, 12]
        - Mean = (4 + 7 + 10 + 5 + 12) / 5 = 38 / 5 = 7.6
    - Practical Usage:
        - Calculating average income in a population
        - Measuring the average test score of a group of students
        - Determining the average temperature over a period of time
    - Diagram/Graph/Plot Used

  - A line plot with data points and a horizontal line showing the mean value.
- Scenario:
  - An online streaming platform wants to understand the average viewing time per user to gauge engagement.
- Problem Statement:
  - The platform needs an overall understanding of how long users spend watching content each day to measure engagement levels.
- Solution:
  - The mean provides the platform with a single value that represents the average time users spend watching. Mean is chosen because it's easy to interpret and provides a quick summary of engagement across all users. The mean works well when data has a fairly symmetric distribution without extreme outliers.
- Alternate Solutions:
  - Median: Useful if there are outliers, like a small group of users watching for significantly longer periods.
  - Mode: Not applicable here, as there may not be a "most common" viewing time.

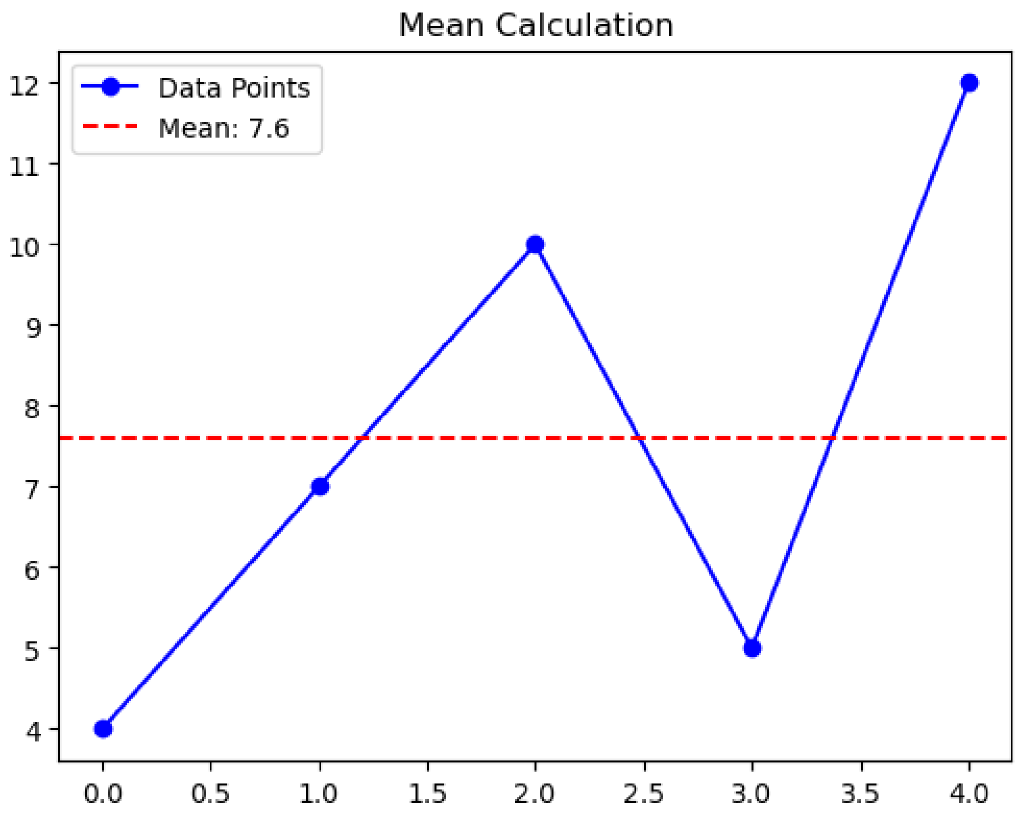
[1]:

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

# Sample data
data = [4, 7, 10, 5, 12]

# Calculate the mean
mean_value = np.mean(data)

# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.axhline(y=mean_value, color='r', linestyle='--', label=f'Mean:␣
  ↪{mean_value}')
plt.title("Mean Calculation")
plt.legend()
plt.show()
```



2. Trimmed Mean
   - Definition: A trimmed mean is a measure of central tendency that involves removing a specified percentage of the highest and lowest values before calculating the mean.
   - Formula:
     - Trimmed Mean = (Σ x_i) / (n - 2k)
     - Where:
       - k = number of data points removed from each end
       - x_i = remaining data points after trimming
       - n = total number of data points
   - Example:
     - For the data set: [4, 7, 10, 5, 12] (removing 1 lowest and 1 highest value)
     - Trimmed data: [7, 10, 5]
     - Trimmed Mean = (7 + 10 + 5) / 3 = 7.33
   - Practical Usage:
     - Robust estimation of the central tendency when data contains extreme outliers.
     - Used in economics to measure central tendency when extreme values distort the results (e.g., income data).
   - Diagram/Graph/Plot Used
     - A line plot showing the original data and the trimmed mean value.
   - Scenario:
     - A retail company wants to assess average monthly spending by customers, excluding

unusually high spenders (e.g., a few corporate clients).

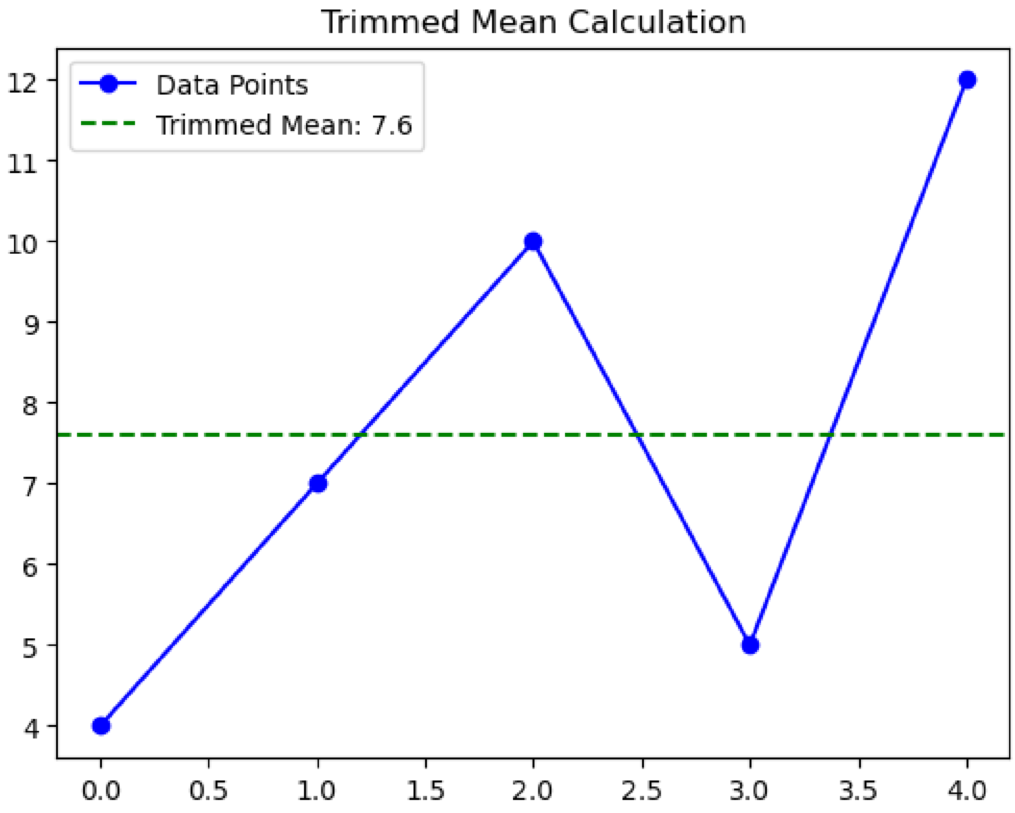
- Problem Statement:
  - Including very high spenders may distort the average and not represent the typical customer’s spending behavior.
- Solution:
  - The trimmed mean calculates the average after removing a percentage of extreme values at both ends, offering a more accurate view of typical spending patterns without the influence of outliers. Trimmed mean is ideal here because it mitigates the skew caused by a few high-spending clients.
- Alternate Solutions:
  - Median: Provides the midpoint of spending, also unaffected by outliers but without taking most data points into account.
  - Mean: Could work if outliers are few, though it’s sensitive to extremes in skewed data.

[2]:

```python
from scipy import stats

# Calculate the trimmed mean (10% trimming)
trimmed_mean_value = stats.trim_mean(data, proportiontocut=0.1)

# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.axhline(y=trimmed_mean_value, color='g', linestyle='--', label=f'Trimmed Mean: {trimmed_mean_value}')
plt.title("Trimmed Mean Calculation")
plt.legend()
plt.show()
```



It appears that 7.6 is showing as the trimmed mean because 10% trimming on this small dataset did not actually remove any values.

- Here's why:
  - When using proportiontocut=0.1, it implies removing 10% from each end.
  - For a dataset with 5 values, 10% trimming would mean removing 0.5 values from each end, which rounds down to 0, resulting intrimming.
  - Since no values are removed, the calculation reverts to the regular mean, which is 7.6.
  - To see a trimmed mean that actually removes values, try a higher proportiontocut, such as 0.2 (which trims 20%, or 1 value from each end)
  - With 0.2 trimming, it will remove one value from each end, resulting in a trimmed mean of 7.33 as calculated from [7,10,5].

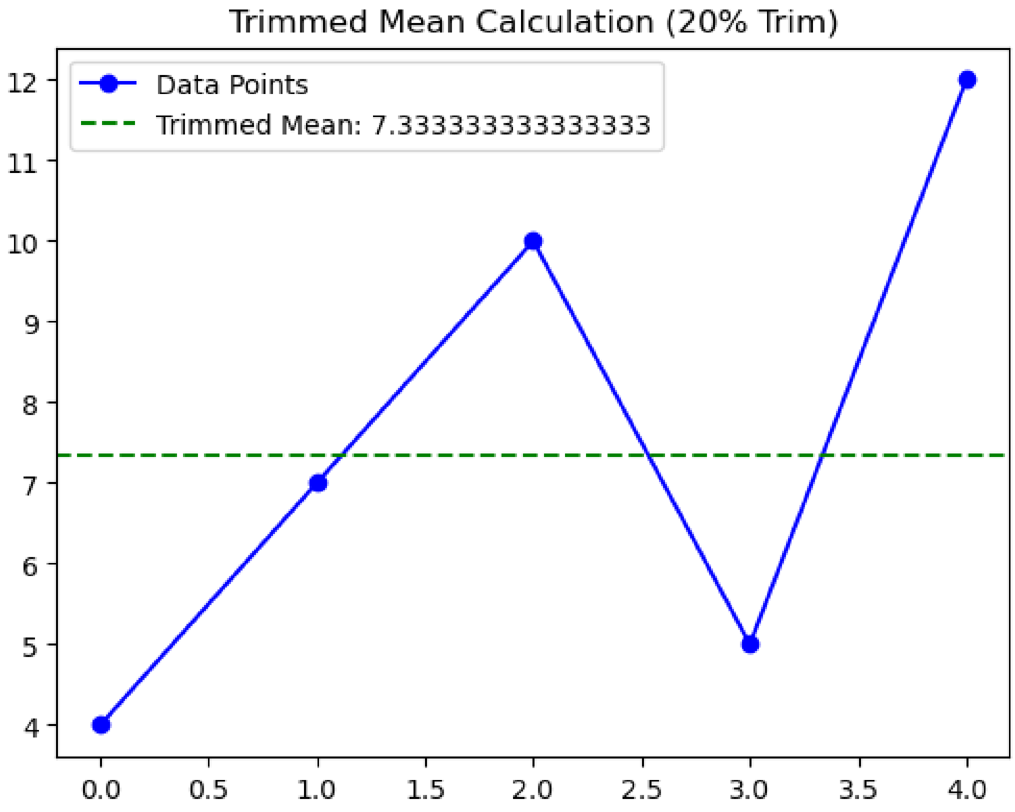
[3]:

```python
from scipy import stats
import matplotlib.pyplot as plt

# Data set
data = [4, 7, 10, 5, 12]

# Calculate the trimmed mean (20% trimming, 10% from each end)
trimmed_mean_value = stats.trim_mean(data, proportiontocut=0.2)
```

```
# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.axhline(y=trimmed_mean_value, color='g', linestyle='--', label=f'Trimmed␣
  ↪Mean: {trimmed_mean_value}')
plt.title("Trimmed Mean Calculation (20% Trim)")
plt.legend()
plt.show()
```



3. Weighted Average
   - Definition: The weighted average is an average where each data point contributes differently based on its weight.
   - Formula:
     - Weighted Average = (Σ x_i * w_i) / Σ w_i
     - Where:
       - x_i = data points
       - w_i = corresponding weights for each data point
   - Example:
     - For the data set: [4, 7, 10, 5, 12] with weights [0.1, 0.3, 0.2, 0.2, 0.2]:
     - Weighted Average = (4 * 0.1 + 7 * 0.3 + 10 * 0.2 + 5 * 0.2 + 12 * 0.2) / (0.1 + 0.3 + 0.2 + 0.2 + 0.2)

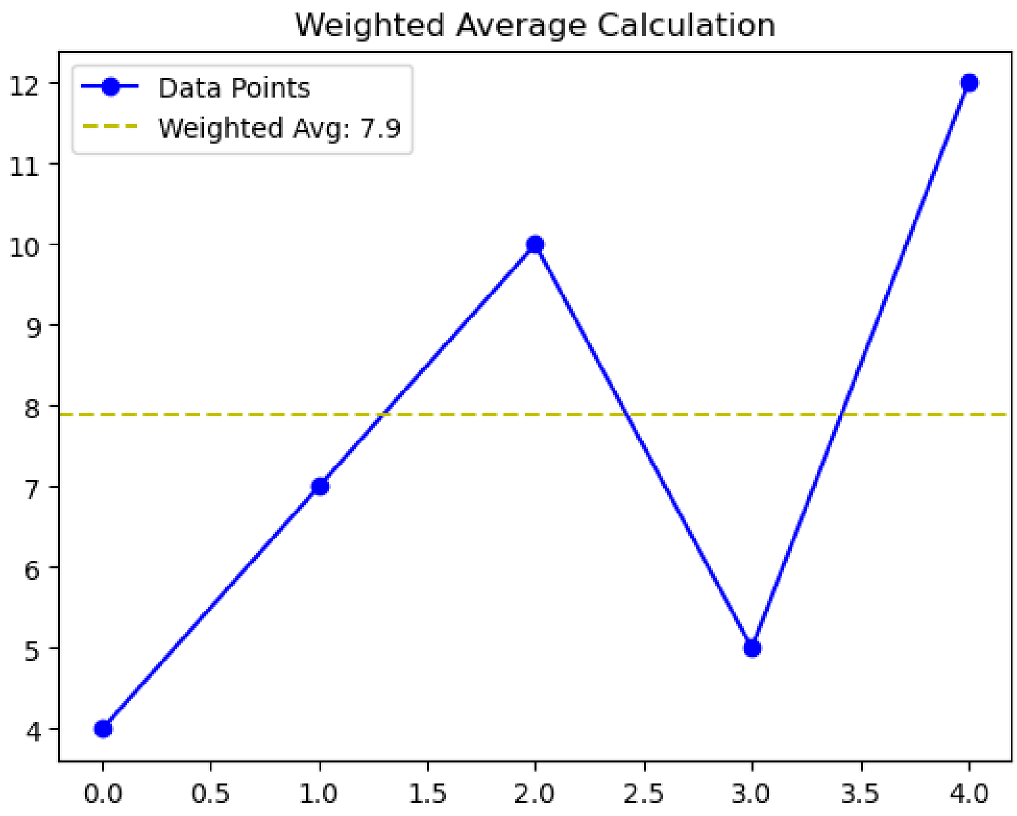
    - Weighted Average = (0.4 + 2.1 + 2 + 1 + 2.4) / 1 = 7.9
- Practical Usage:
    - Used when certain data points are more important than others.
    - Applications in finance (e.g., weighted portfolio return) or education (e.g., weighted average of grades).
- Diagram/Graph/Plot Used
    - A line plot with weighted average shown as a horizontal line.
- Scenario:
    - A university calculates students' final grades using a weighted system where exams are worth 60% and assignments are worth 40%.
- Problem Statement:
    - Each component of the grade has a different importance, and a simple average won't correctly represent each student's overall performance.
- Solution:
    - The weighted average accounts for the varying importance of each grade component, giving exams more influence than assignments. It's the most appropriate solution here, as it directly reflects the relative importance assigned to each component of the overall grade.
- Alternate Solutions:
    - Mean: Useful if all components (exams, assignments) had equal importance, which isn't the case here.
    - Median: Unfit for this scenario, as it would ignore the weight assigned to each component.

[4]:

```python
# Weights corresponding to the data
weights = [0.1, 0.3, 0.2, 0.2, 0.2]

# Calculate the weighted average
weighted_avg = np.average(data, weights=weights)

# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.axhline(y=weighted_avg, color='y', linestyle='--', label=f'Weighted Avg: {weighted_avg}')
plt.title("Weighted Average Calculation")
plt.legend()
plt.show()
```



4. Mode
   - Definition: The mode is the value that occurs most frequently in a data set.
   - Example:
     - For the data set: [4, 7, 10, 5, 12, 7]
     - Mode = 7 (because it appears twice)
   - Practical Usage:
     - Commonly used in categorical data (e.g., mode of favorite colors).
     - Useful in retail to find the most popular products.
   - Diagram/Graph/Plot Used
     - Histogram to visualize the frequency of data points, with a vertical line indicating the mode.
   - Scenario:
     - A grocery store analyzes the most common products sold during the weekend to ensure they stock popular items.
   - Problem Statement:
     - The store needs to know which products are purchased most frequently over the weekend to manage inventory and avoid stockouts.
   - Solution:
     - The mode identifies the most frequently purchased items, enabling the store to focus on stocking items in high demand. The mode is ideal because it directly shows the most popular choice among categorical options (product types), which is critical in

retail for demand forecasting.

- Alternate Solutions:
  - Mean or Median Sales: Not applicable here as they provide average sales, which doesn't reveal specific product popularity.
  - Percentiles: Could be used to rank products based on demand, though it wouldn't directly pinpoint the most frequently sold items.

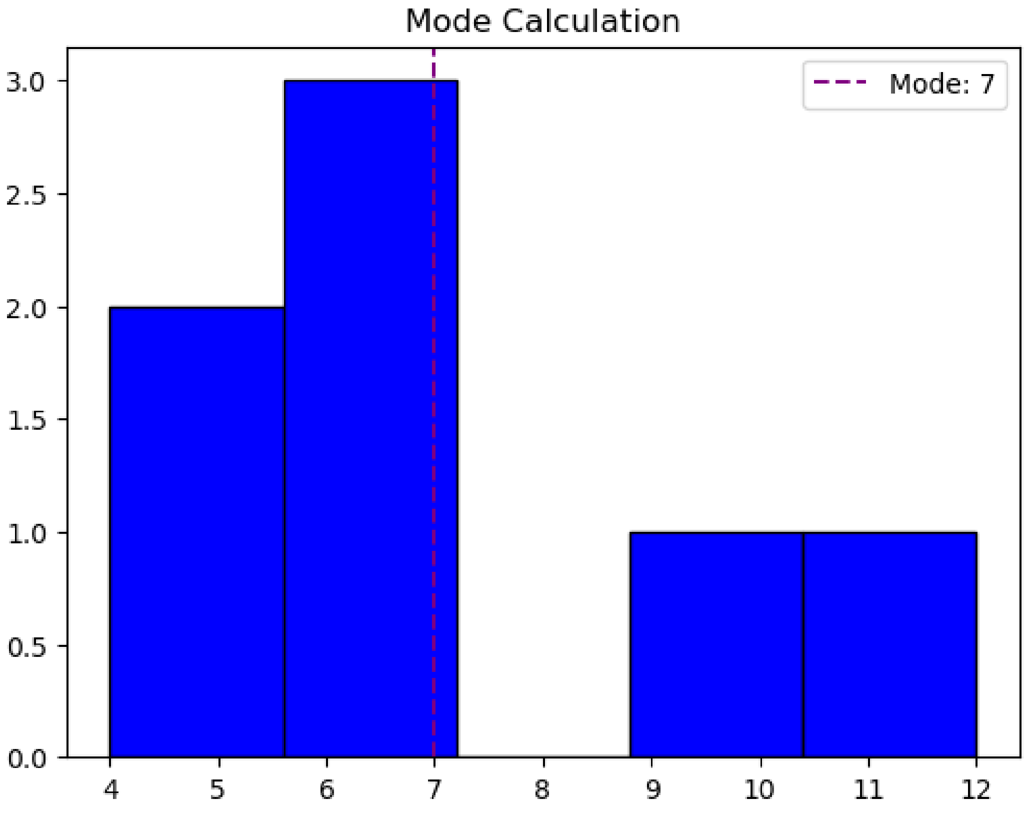
[5]:

```python
from scipy import stats
import matplotlib.pyplot as plt

# Sample data
data = [4, 7, 10, 5, 7, 12, 7]

# Calculate mode
mode_value = stats.mode(data, keepdims=True).mode[0]  # Using keepdims=True for compatibility

# Plot the data
plt.hist(data, bins=len(set(data)), color='b', edgecolor='black')
plt.axvline(x=mode_value, color='purple', linestyle='--', label=f'Mode: {mode_value}')
plt.title("Mode Calculation")
plt.legend()
plt.show()
```

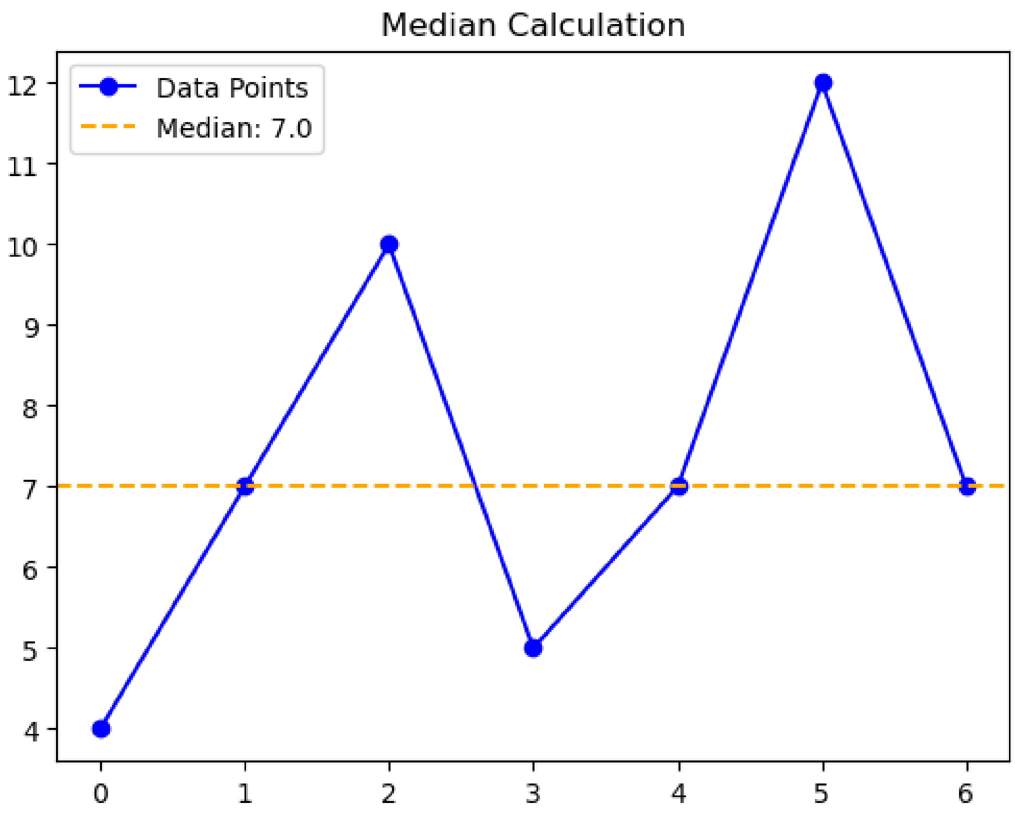


5. Median
    - Definition: The median is the middle value in a data set when arranged in ascending order.
    - Formula:
        1. Sort the data.
        2. If the data set has an odd number of points, the median is the middle value.
        3. If the data set has an even number of points, the median is the average of the two middle values.
        4. Median = [(n + 1)/2]th term if the number of observations (n) is odd, and Median = [(n/2)th term + (n/2 + 1)th term] / 2 if the number of observations is even
    - Example:
        - For the data set: [4, 7, 10, 5, 12] (sorted: [4, 5, 7, 10, 12])
        - Median = 7
    - Practical Usage:
        - The median is more robust than the mean in the presence of outliers.
        - Often used in income distribution or house prices.
    - Diagram/Graph/Plot Used
        - Line plot with the median shown as a horizontal line.
    - Scenario:
        - A real estate company needs to determine the typical price of homes in an area with a few extremely high-priced properties.
    - Problem Statement:
        - The mean house price is skewed due to a few luxury homes, so the company needs a measure that better represents a typical home price for regular buyers.
    - Solution:
        - Median provides the midpoint of home prices, unaffected by outliers, giving a better sense of the “typical” home cost for most buyers.
        - Median is ideal here as it remains stable in the presence of extreme values, which is common in real estate markets.
    - Alternate Solutions:
        - Mean: Would be misleading due to outliers in luxury properties.
        - Mode: Could show the most common price, but home prices often vary too much for mode to be meaningful.

[6]:

```python
# Calculate median
median_value = np.median(data)

# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.axhline(y=median_value, color='orange', linestyle='--', label=f'Median: {median_value}')
plt.title("Median Calculation")
plt.legend()
plt.show()
```



6. Outliers
   - Definition: Outliers are data points that are significantly different from the rest of the data. They can distort statistical analyses.
   - Formula (IQR Method):
     - Outliers = x_i < Q1 - 1.5 * IQR or x_i > Q3 + 1.5 * IQR
     - Where:
       - Q1 = First quartile
       - Q3 = Third quartile
       - IQR = Interquartile Range (Q3 - Q1)
   - Example: For the data set: [4, 7, 10, 5, 100] Calculate Q1, Q3, and IQR. The value 100 is an outlier.
   - Practical Usage:
     - Used to identify erroneous or extreme data points in business or scientific data.
   - Diagram/Graph/Plot Used
     - Line plot with horizontal lines showing the upper and lower bounds for outliers.
   - Scenario:
     - A healthcare provider reviews patient stay durations to spot unusual cases for quality assurance.
   - Problem Statement:
     - Most patients stay for 3-5 days, but some outliers stay for significantly longer, possibly indicating complications or special circumstances.

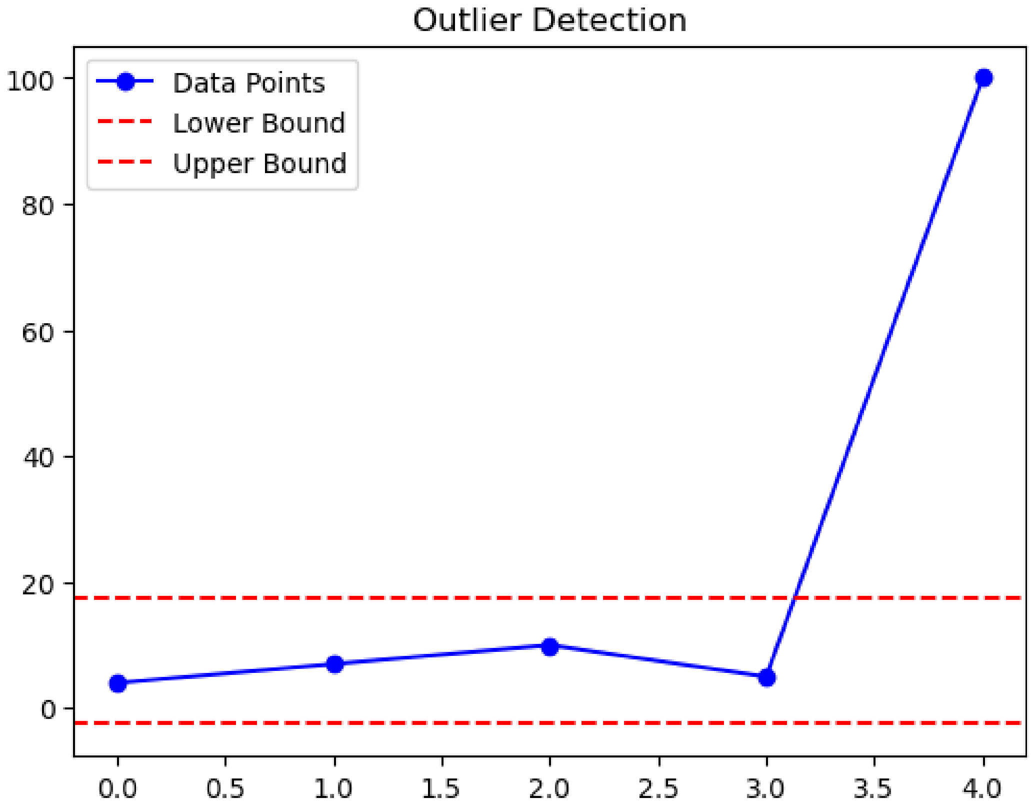
- Solution:
  - Identifying outliers helps the provider investigate specific cases that deviate from the typical stay duration, which can help identify quality of care issues or special cases requiring further attention. Outliers are important here because they reveal cases that warrant further investigation and can drive quality improvements.
- Alternate Solutions:
  - Interquartile Range (IQR): A useful method for determining the range within which typical data falls, identifying values outside the IQR as potential outliers.
  - Standard Deviation Method: Outliers could also be identified using standard deviation, though it's less robust in highly skewed data.

[7]:

```
# Detecting outliers using IQR
data = [4, 7, 10, 5, 100]
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print("Outliers:", outliers)
# Plot the data
plt.plot(data, 'bo-', label="Data Points")
plt.title("Outlier Detection")
plt.axhline(y=lower_bound, color='r', linestyle='--', label='Lower Bound')
plt.axhline(y=upper_bound, color='r', linestyle='--', label='Upper Bound')
plt.legend()
plt.show()
```

```
Outliers: [100]
```


Outlier Detection
Data Points
Lower Bound
Upper Bound
100
80
60
40
20
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0