Statistik-LÖS-Lage-Streumasse

August 28, 2024

1 Teil 1: Lagemasse

Dieses Notebook enthält eine Serie von 10 Statistik-Aufgaben, die sich auf Lagemasse und Streuungsmasse beziehen. Jede Aufgabe wird mit einer theoretischen Erklärung, den relevanten Formeln, einem Datensatz und einer Lösung (inklusive Visualisierung) dargestellt.

1.1 Aufgabe 1: Berechne den Mittelwert und visualisiere die Daten

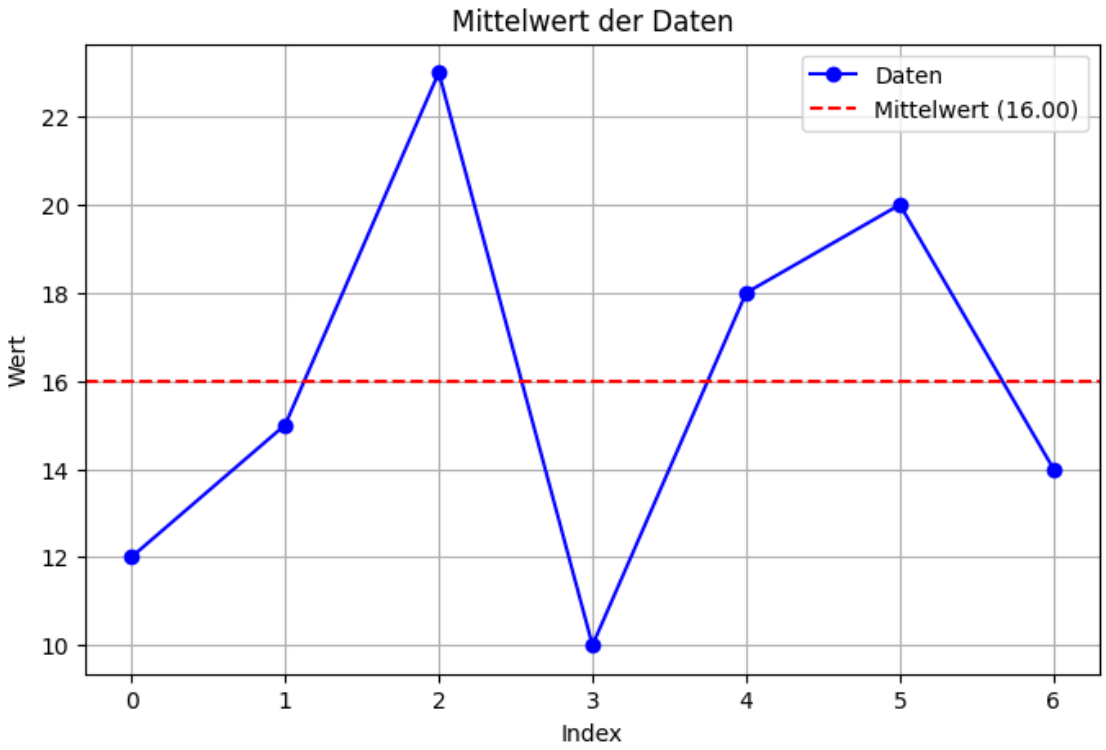
Theorie: Der Mittelwert (arithmetisches Mittel) ist das am häufigsten verwendete Maß für die zentrale Tendenz. Er wird berechnet, indem man die Summe aller Werte durch die Anzahl der Werte teilt.

Formel:

$$\text{Mittelwert} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Datensatz: `[12, 15, 23, 10, 18, 20, 14]`

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

data = [12, 15, 23, 10, 18, 20, 14]
mean_value = np.mean(data)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'bo-', label="Daten")
plt.axhline(y=mean_value, color='r', linestyle='--', label=f'Mittelwert␣
 ↪({mean_value:.2f})')
plt.title("Mittelwert der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()
```



1.2 Aufgabe 2: Bestimme den Median und visualisiere die Daten

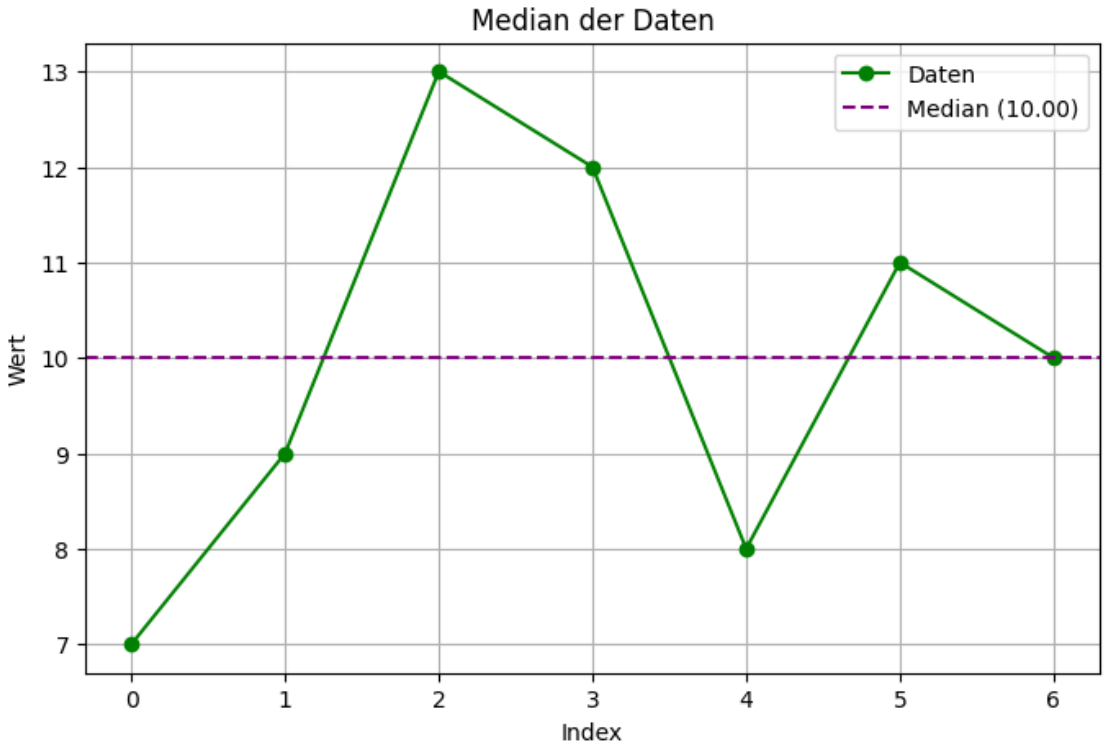
Theorie: Der Median ist der mittlere Wert eines geordneten Datensatzes und teilt die Daten in zwei gleich große Hälften. Er ist robuster gegenüber Ausreißern als der Mittelwert.

Formel: - Für ungerade Anzahl an Werten: Der Median ist der mittlere Wert. - Für gerade Anzahl an Werten: Der Median ist der Durchschnitt der beiden mittleren Werte.

Datensatz: `[7, 9, 13, 12, 8, 11, 10]`

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

data = [7, 9, 13, 12, 8, 11, 10]
median_value = np.median(data)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'go-', label="Daten")
plt.axhline(y=median_value, color='purple', linestyle='--', label=f'Median␣
 ↪({median_value:.2f})')
plt.title("Median der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
```

```
plt.legend()
plt.grid(True)
plt.show()
```



1.3 Aufgabe 3: Finde den Modus und visualisiere die Daten

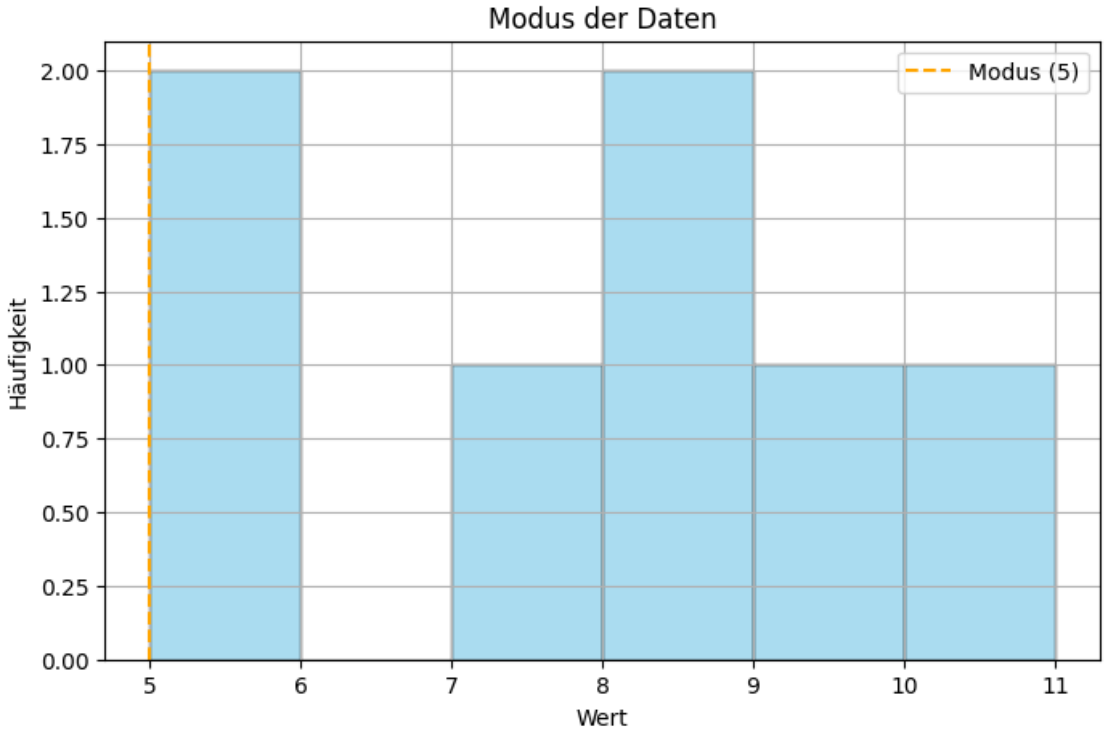
Theorie: Der Modus ist der Wert, der in einem Datensatz am häufigsten vorkommt. Es kann mehr als einen Modus geben, wenn mehrere Werte die gleiche Häufigkeit haben.

Formel: - Der Modus ist der Wert mit der höchsten Häufigkeit im Datensatz.

Datensatz: `[5, 8, 9, 8, 10, 5, 7]`

[ ]:
```
from scipy import stats
import matplotlib.pyplot as plt

data = [5, 8, 9, 8, 10, 5, 7]
mode_value = stats.mode(data, keepdims=True)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.hist(data, bins=range(min(data), max(data) + 2), alpha=0.7,␣
↪color='skyblue', edgecolor='black')
```

```
plt.axvline(x=mode_value.mode[0], color='orange', linestyle='--', label=f'Modus␣
↪({mode_value.mode[0]})')
plt.title("Modus der Daten")
plt.xlabel("Wert")
plt.ylabel("Häufigkeit")
plt.legend()
plt.grid(True)
plt.show()
```



1.4 Aufgabe 4: Berechne den gewichteten Mittelwert und visualisiere die Daten

Theorie: Der gewichtete Mittelwert ist eine Variante des Mittelwerts, bei der einige Datenpunkte mehr Einfluss (Gewicht) haben als andere. Er wird häufig verwendet, wenn bestimmte Werte wichtiger sind als andere.

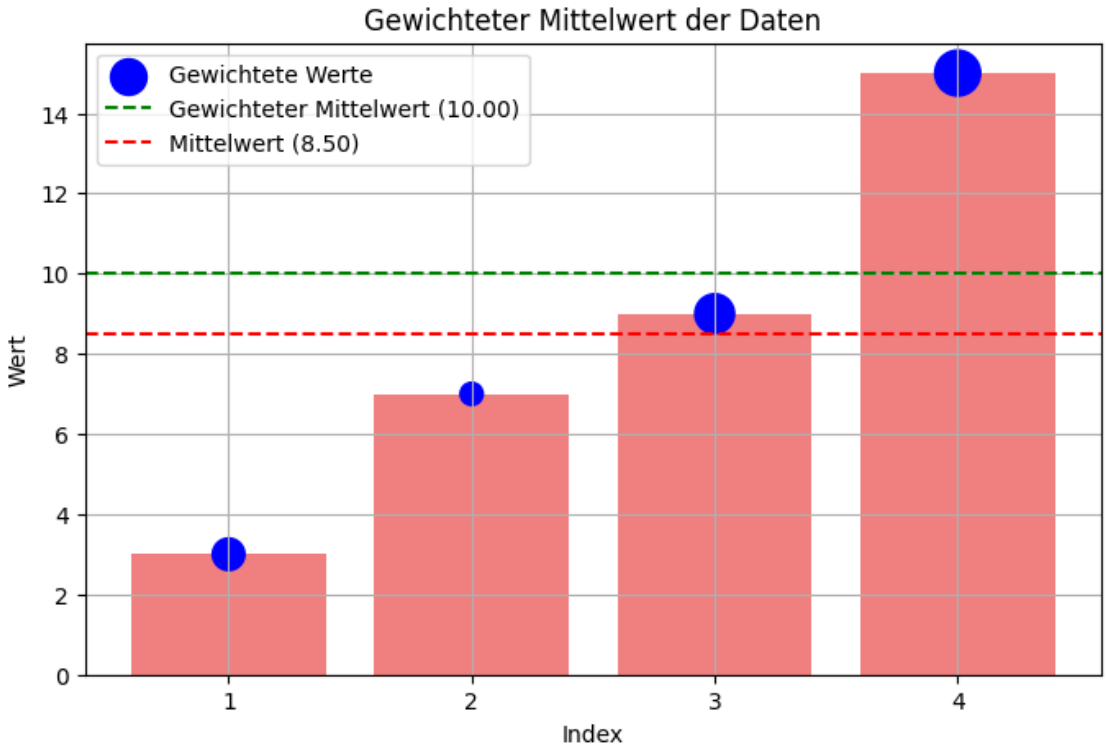
Formel:

$$\text{Gewichteter Mittelwert} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

wobei (w_i) die Gewichte und (x_i) die Datenwerte sind.

Datensatz: Werte: `[3, 7, 9, 15]`, Gewichte: `[2, 1, 3, 4]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [3, 7, 9, 15]
weights = [2, 1, 3, 4]
weighted_mean = np.average(data, weights=weights)
mittel = np.average(data)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.bar(range(len(data)), data, color='lightcoral', tick_label=range(1,␣
 ↪len(data) + 1))
plt.scatter(range(len(data)), data, s=np.array(weights) * 100, color='blue',␣
 ↪label="Gewichtete Werte")
plt.axhline(y=weighted_mean, color='green', linestyle='--', label=f'Gewichteter␣
 ↪Mittelwert ({weighted_mean:.2f})')

plt.axhline(y=mittel, color='red', linestyle='--', label=f'Mittelwert ({mittel:.
 ↪2f})')



plt.title("Gewichteter Mittelwert der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()
```



1.5 Aufgabe 5: Berechne das arithmetische Mittel und den Median und visualisiere die Daten

Theorie: Der Mittelwert und der Median sind beides Lagemaße, aber sie reagieren unterschiedlich auf Ausreißer. Der Mittelwert wird durch extreme Werte stärker beeinflusst, während der Median robuster gegenüber Ausreißern ist.

Formeln: - **Mittelwert:**

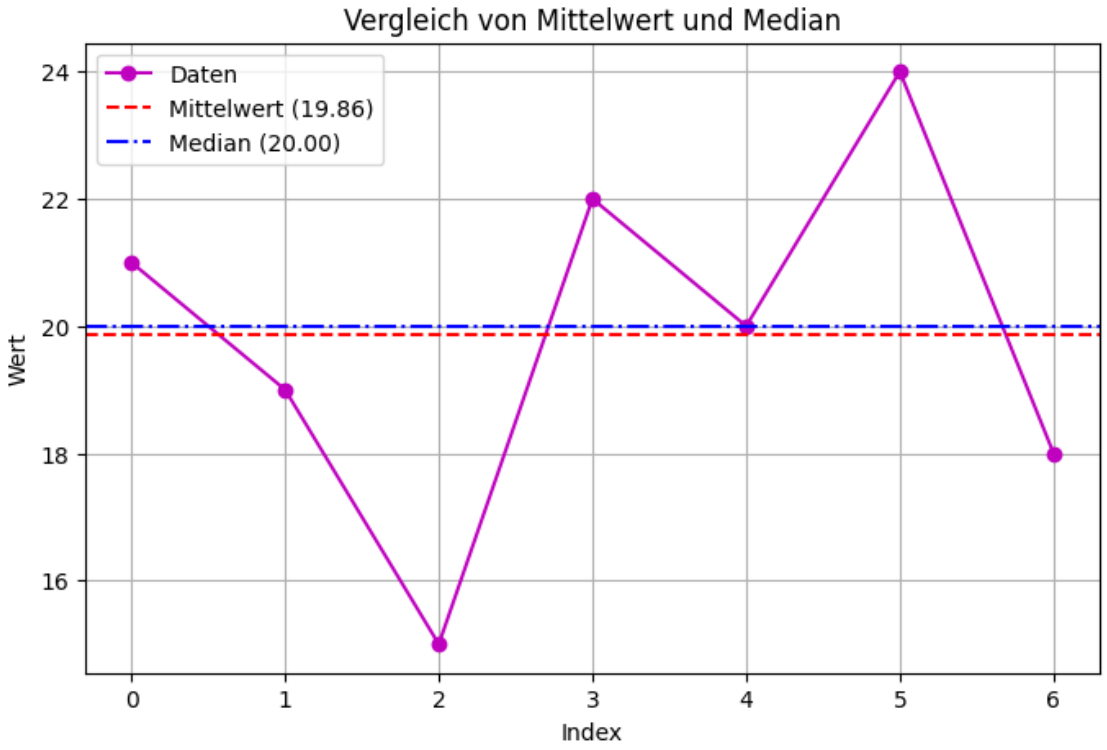
$$\text{Mittelwert} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- **Median:** Der mittlere Wert in einem geordneten Datensatz.

Datensatz: `[21, 19, 15, 22, 20, 24, 18]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [21, 19, 15, 22, 20, 24, 18]
mean_value = np.mean(data)
median_value = np.median(data)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'mo-', label="Daten")
```

```
plt.axhline(y=mean_value, color='r', linestyle='--', label=f'Mittelwert␣
 ↪({mean_value:.2f})')
plt.axhline(y=median_value, color='b', linestyle='-.', label=f'Median␣
 ↪({median_value:.2f})')
plt.title("Vergleich von Mittelwert und Median")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()
```



2 Teil 2: Streuungsmasse

2.1 Aufgabe 6: Berechne die Spannweite und visualisiere die Daten

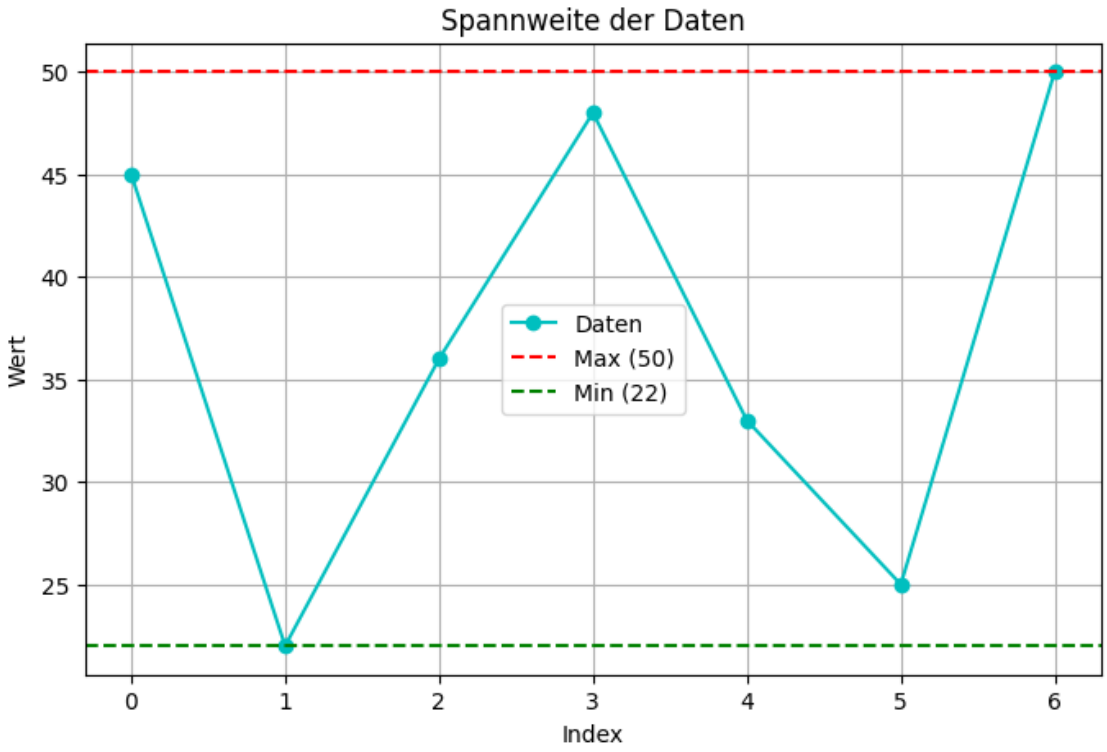
Theorie: Die Spannweite ist das einfachste Streuungsmaß und wird berechnet, indem man den kleinsten Wert vom größten Wert des Datensatzes subtrahiert. Sie gibt die Gesamtstreuung der Daten an.

Formel:

$$\text{Spannweite} = \text{Max} - \text{Min}$$

Datensatz: `[45, 22, 36, 48, 33, 25, 50]`

```
[ ]: data = [45, 22, 36, 48, 33, 25, 50]
range_value = max(data) - min(data)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'co-', label="Daten")
plt.axhline(y=max(data), color='r', linestyle='--', label=f'Max ({max(data)})')
plt.axhline(y=min(data), color='g', linestyle='--', label=f'Min ({min(data)})')
plt.title("Spannweite der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()

print("Min:", min(data), "Max:", max(data), "Spannweite:", range_value)
```




```
Min: 22 Max: 50 Spannweite: 28
```

2.2 Aufgabe 7: Berechne die Varianz und visualisiere die Daten

Theorie: Die Varianz ist ein Mass dafür, wie weit die einzelnen Werte eines Datensatzes im Durchschnitt vom Mittelwert entfernt sind. Sie wird berechnet, indem man die Abweichungen der

einzelnen Werte vom Mittelwert quadriert und den Durchschnitt dieser Quadrate bildet.

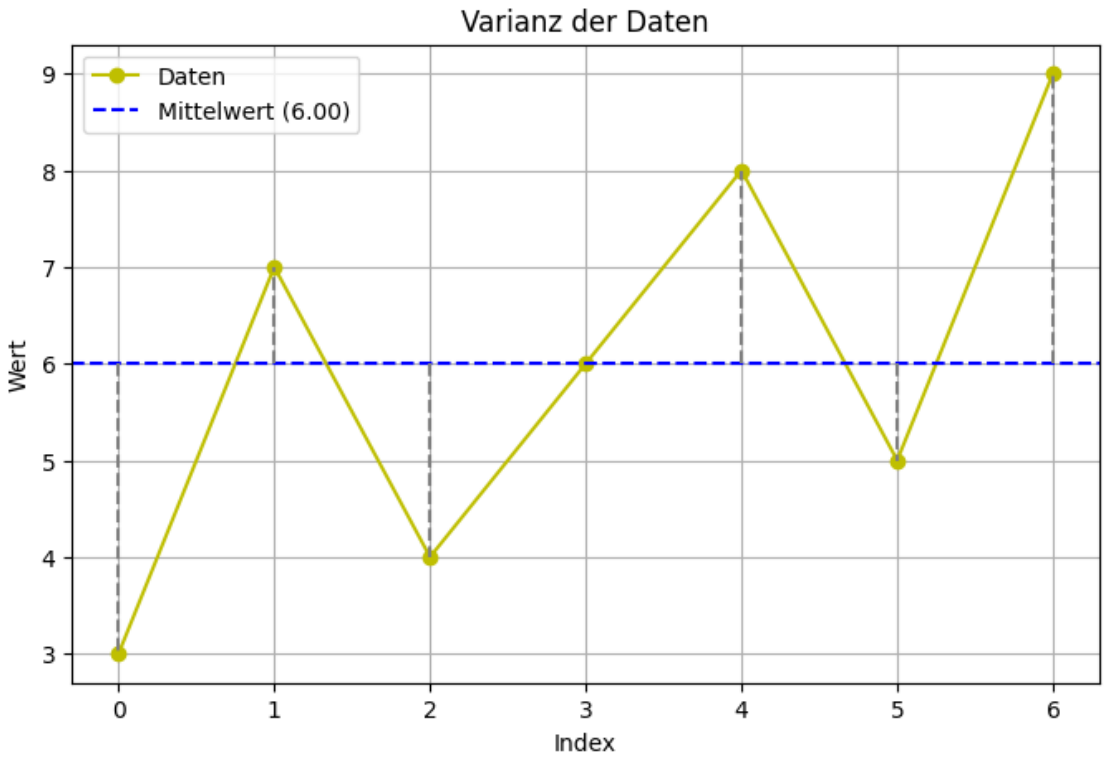
Formel:

$$\text{Varianz} = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

wobei (μ) der Mittelwert ist.

Datensatz: `[3, 7, 4, 6, 8, 5, 9]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [3, 7, 4, 6, 8, 5, 9]
variance = np.var(data, ddof=0)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'yo-', label="Daten")
plt.axhline(y=np.mean(data), color='b', linestyle='--', label=f'Mittelwert ({np.
 ↪mean(data):.2f})')
for i, value in enumerate(data):
    plt.vlines(i, np.mean(data), value, color='gray', linestyle='--')
plt.title("Varianz der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()

print("Varianz:", variance)
```




```
Varianz: 4.0
```

2.3 Aufgabe 8a: Berechne die Standardabweichung und visualisiere die Daten

Theorie: Die Standardabweichung ist die Quadratwurzel der Varianz und gibt die durchschnittliche Abweichung der Datenwerte vom Mittelwert an. Sie ist ein gängiges Mass für die Streuung der Daten.

Formel:

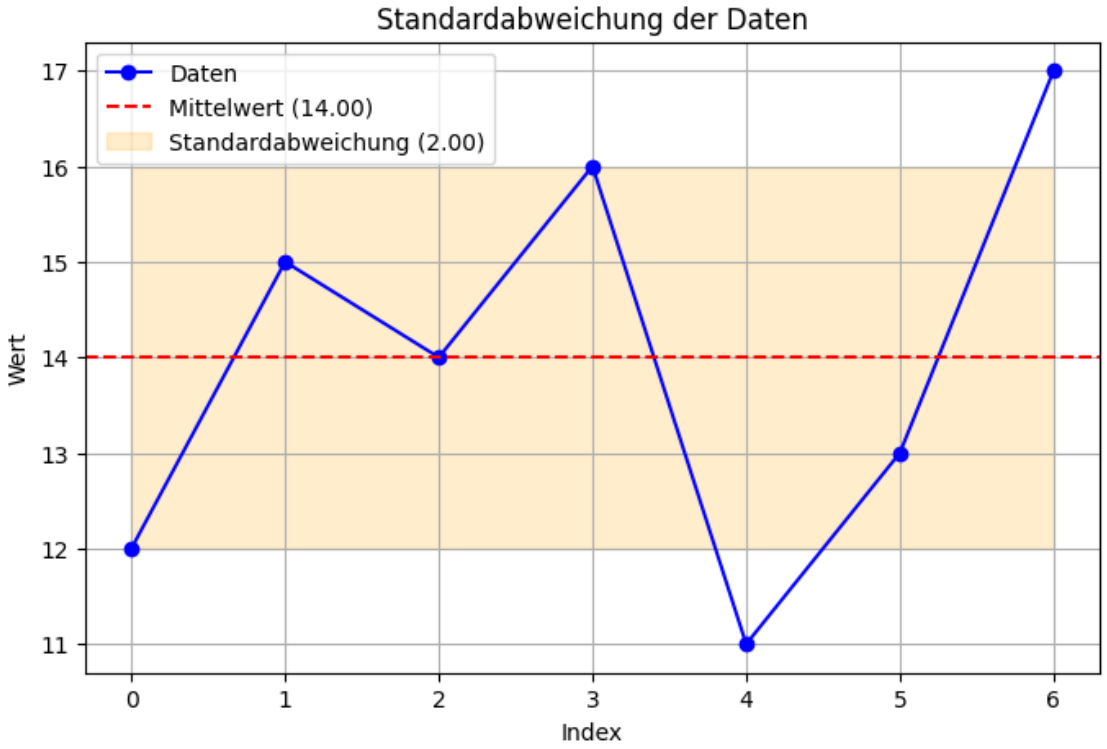
$$\text{Standardabweichung} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

wobei (μ) der Mittelwert ist.

Datensatz: `[12, 15, 14, 16, 11, 13, 17]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [12, 15, 14, 16, 11, 13, 17]
std_dev = np.std(data, ddof=0)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'bo-', label="Daten")
```

```
plt.axhline(y=np.mean(data), color='r', linestyle='--', label=f'Mittelwert ({np.
  ↪mean(data):.2f})')
plt.fill_between(range(len(data)), np.mean(data) - std_dev, np.mean(data) +␣
  ↪std_dev, color='orange', alpha=0.2, label=f'Standardabweichung ({std_dev:.
  ↪2f})')
plt.title("Standardabweichung der Daten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()

print("Standardabweichung:", std_dev)
```




```
Standardabweichung: 2.0
```

2.4 Aufgabe 8b: Standardabweichung für Stichprobendaten berechnen

Theorie: Die Standardabweichung einer Stichprobe wird ähnlich wie die Standardabweichung der gesamten Population berechnet, jedoch mit einer kleinen Anpassung im Nenner. Diese Anpassung, bei der man im Nenner 1 vom Stichprobenumfang subtrahiert, wird als Bessel-Korrektur bezeichnet. Diese Korrektur wird durchgeführt, um eine Verzerrung in der Schätzung der Varianz und der Standardabweichung zu vermeiden.

Formel:

$$\text{Standardabweichung der Stichprobe} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

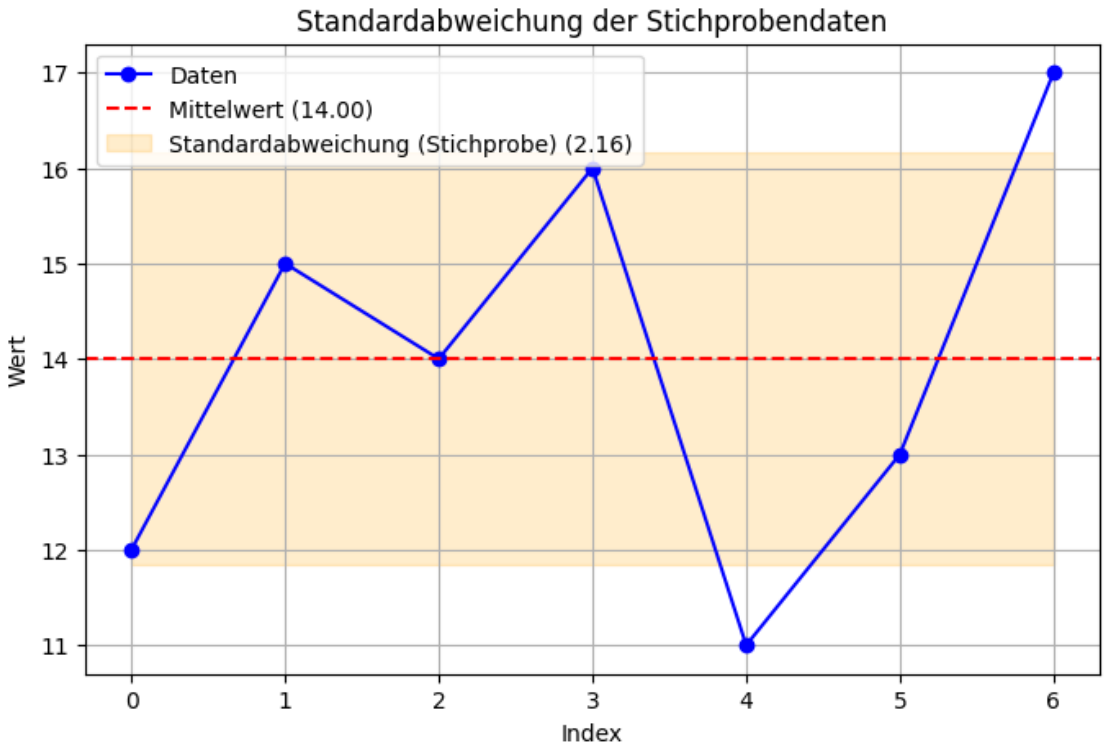
wobei ($\bar{x}$) der Stichprobenmittelwert ist.

Warum wird 1 im Nenner subtrahiert?

Der Grund für die Subtraktion von 1 im Nenner (also die Verwendung von (n-1) anstelle von (n)) liegt darin, dass eine Stichprobe eine unvollständige Abbildung der Population ist. Wenn man den Mittelwert einer Stichprobe berechnet, nähert er sich tendenziell dem tatsächlichen Populationsmittelwert an, aber er ist nicht exakt. Diese Korrektur sorgt dafür, dass die Berechnung der Varianz (und damit auch der Standardabweichung) nicht systematisch unterschätzt wird. Durch das Dividieren durch (n-1) wird die Schätzung der Varianz und der Standardabweichung „entzerrt“, um genauer die Populationsparameter widerzuspiegeln.

Datensatz: `[12, 15, 14, 16, 11, 13, 17]`

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

# Datensatz
data = [12, 15, 14, 16, 11, 13, 17]

# Berechnung der Standardabweichung für Stichproben (mit Bessel-Korrektur)
std_dev_sample = np.std(data, ddof=1)

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'bo-', label="Daten")
plt.axhline(y=np.mean(data), color='r', linestyle='--', label=f'Mittelwert ({np.
 ↪mean(data):.2f})')
plt.fill_between(range(len(data)), np.mean(data) - std_dev_sample, np.
 ↪mean(data) + std_dev_sample, color='orange', alpha=0.2,␣
 ↪label=f'Standardabweichung (Stichprobe) ({std_dev_sample:.2f})')
plt.title("Standardabweichung der Stichprobendaten")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()

print("Standardabweichung der Stichprobe:", std_dev_sample)
```




```
Standardabweichung der Stichprobe: 2.160246899469287
```

2.5 Aufgabe 9: Berechne den Quartilsabstand und visualisiere die Daten

Theorie: Der Quartilsabstand (Interquartilsabstand, IQR) ist ein Streuungsmaß, das die Spannweite der mittleren 50% der Daten angibt. Er wird berechnet, indem man das erste Quartil vom dritten Quartil subtrahiert.

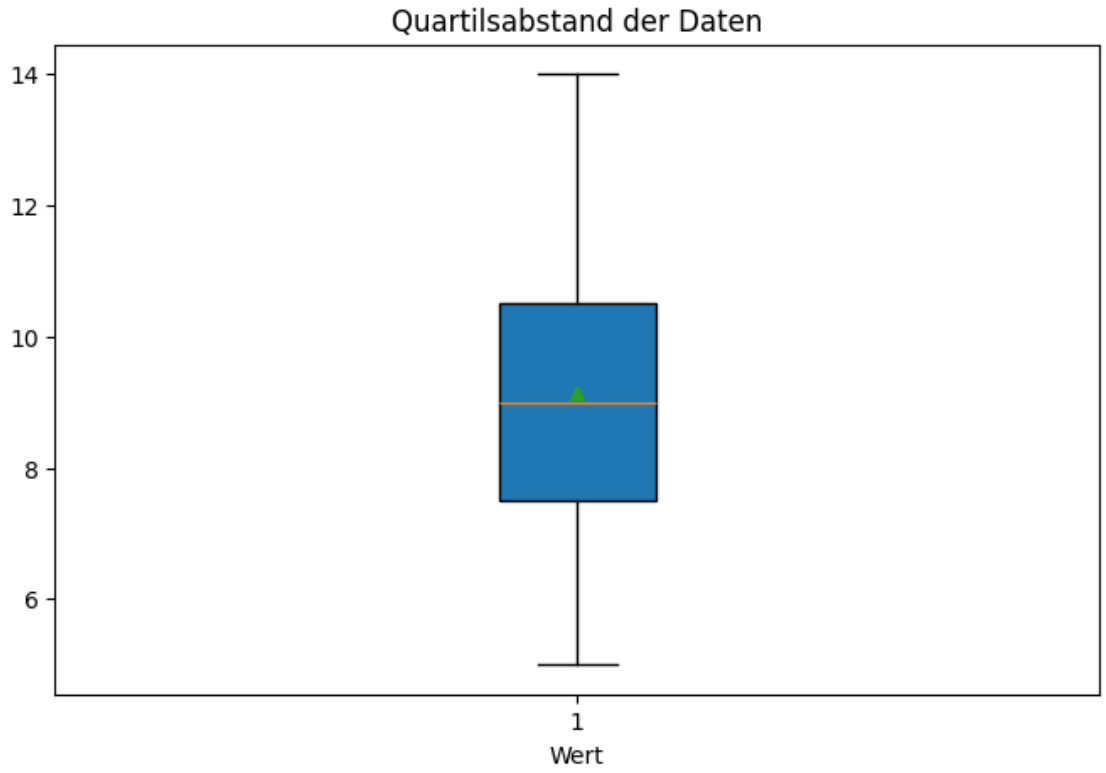
Formel:

$$IQR = Q3 - Q1$$

Datensatz: `[5, 9, 14, 7, 11, 8, 10]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [5, 9, 14, 7, 11, 8, 10]
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1

# Visualisierung
plt.figure(figsize=(8, 5))
plt.boxplot(data, vert=True, patch_artist=True, showmeans=True)
```

```
plt.title("Quartilsabstand der Daten")
plt.xlabel("Wert")
plt.show()

print("Quartilsabstand (IQR):", iqr)
```




```
Quartilsabstand (IQR): 3.0
```

2.6 Aufgabe 10: Berechne die mittlere absolute Abweichung und visualisiere die Daten

Theorie: Die mittlere absolute Abweichung (MAD) ist das durchschnittliche absolute Maß, wie weit die Werte eines Datensatzes von ihrem Mittelwert entfernt sind. Sie ist ein alternatives Streuungsmaß zur Standardabweichung.

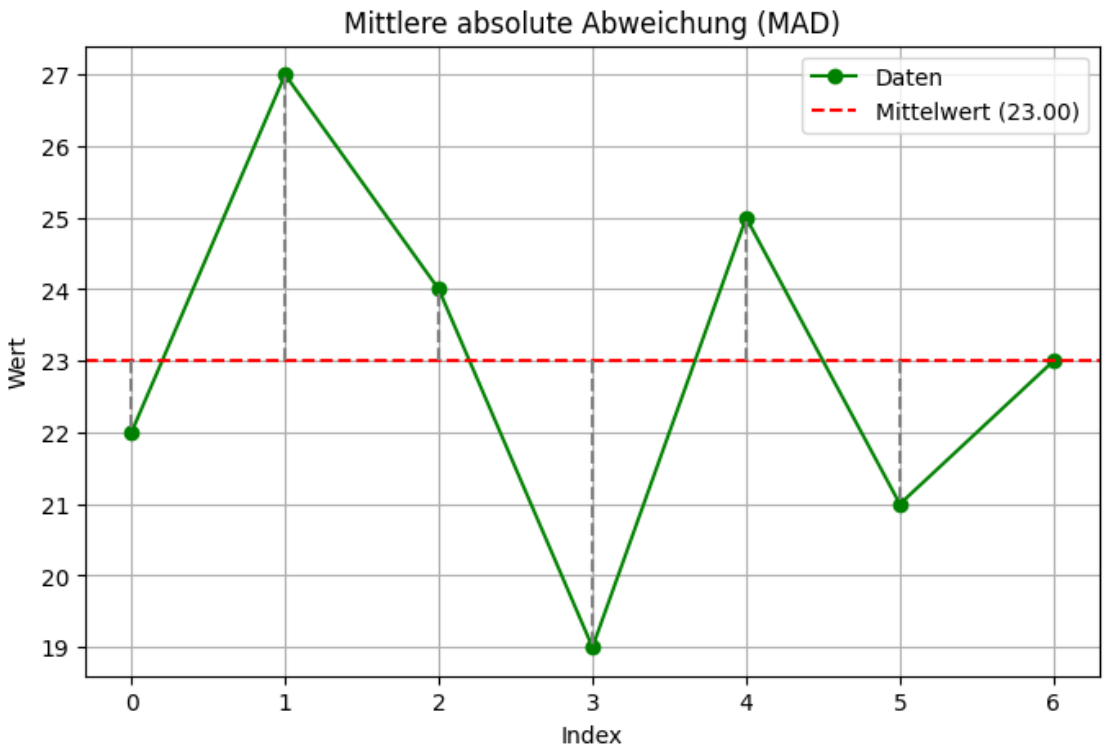
Formel:

$$MAD = \frac{\sum_{i=1}^{n} |x_i - \mu|}{n}$$

wobei ( ) der Mittelwert ist.

Datensatz: `[22, 27, 24, 19, 25, 21, 23]`

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = [22, 27, 24, 19, 25, 21, 23]
mad = np.mean(np.abs(data - np.mean(data)))

# Visualisierung
plt.figure(figsize=(8, 5))
plt.plot(data, 'go-', label="Daten")
plt.axhline(y=np.mean(data), color='r', linestyle='--', label=f'Mittelwert ({np.
 ↪mean(data):.2f})')
for i, value in enumerate(data):
    plt.vlines(i, np.mean(data), value, color='gray', linestyle='--')
plt.title("Mittlere absolute Abweichung (MAD)")
plt.xlabel("Index")
plt.ylabel("Wert")
plt.legend()
plt.grid(True)
plt.show()

print("Mittlere absolute Abweichung (MAD):", mad)
```




```
Mittlere absolute Abweichung (MAD): 2.0
```