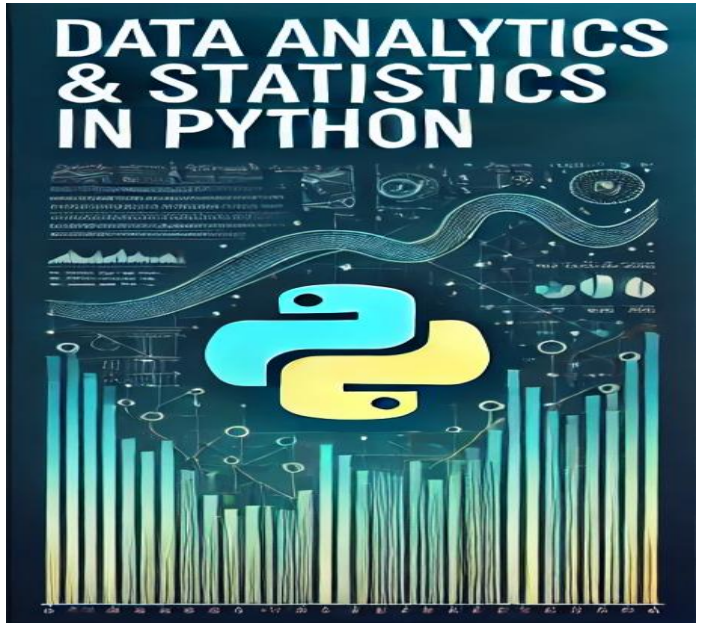
Data Analytics & Statistics in Python
Session 3: Statistics in Python



Metropolia
University of Applied Sciences

Learning data-driven decision-making with Python

Instructor: Hamed Ahmadinia, Ph.D.

Email: hamed.ahmadinia@metropolia.fi

Concepts of Today

- **Key Topics Covered Today:**
  - **Minimum and Maximum:**
    - Values:min(iterable), max(iterable) — Find smallest and largest values.
    - NumPy: np.min(), np.max() — Works on arrays and supports axis-based operations.
    - Pandas: df.min(), df.max() — Apply on DataFrame columns or rows.
  - **Means of Values (Arithmetic, Geometric, Harmonic, Weighted):**
    - sum(data)/len(data) — Arithmetic mean (simple average).
    - NumPy: np.mean(data) — Faster for large datasets.
    - scipy.stats.gmean(data) — Geometric mean (growth rates).
    - scipy.stats.hmean(data) — Harmonic mean (rates).
    - Weighted Mean: np.average(data, weights=weights) — Weighted by importance.
  - **Median and Mode:**
    - statistics.median(data) / np.median(data) — Median (middle value).
    - scipy.stats.mode(data) — Mode (most common value).
  - **Quantiles and IQR:**
    - np.quantile(data, [0.25, 0.5, 0.75]) — Calculate quartiles.
    - IQR: Q3 - Q1 — Range of the middle 50% of data.
    - Boxplot: plt.boxplot(data) — Visualize IQR and detect outliers.

Minimum and Maximum Values

- **Why Are Minimum and Maximum Values Important?**
  - Help calculate range: difference between the maximum and minimum values.
  - Used to normalize data (scale values between 0 and 1).
  - Detect outliers: values that are unusually high or low.
  - Set limits for data visualizations.

```
# Sample sales data
data = {'Country': ['USA', 'USA', 'Canada', 'USA', 'Mexico', 'Canada'],
        'Sales': [200, 150, 300, 400, 250, 100]}

# Create DataFrame
df = pd.DataFrame(data)

# Filter sales in the USA and get min/max
usa_sales = df[df['Country'] == 'USA']
min_sales = usa_sales['Sales'].min()
max_sales = usa_sales['Sales'].max()

print(f"Minimum Sales in USA: {min_sales}")
print(f"Maximum Sales in USA: {max_sales}")
```

```
Minimum Sales in USA: 150
Maximum Sales in USA: 400
```

Finding Min/Max in Python

- **Lists and NumPy Arrays**
  - Python's built-in functions: min(), max()
  - NumPy for larger data: np.min(), np.max() (faster and supports arrays)
- **Pandas DataFrames**
  - Min/Max for columns/rows: df.min(), df.max()
- **Grouping Data**
  - Group by categories and apply min/max

```python
numbers = [1, 2, 3, 4, -5]
print(min(numbers), max(numbers))  # Output: -5, 4

arr = np.array([[1, 2, 3], [4, 5, 6]])
print(np.min(arr, axis=1))  # Row-wise min: [1, 4]

# 2. Pandas DataFrame
import pandas as pd
df = pd.DataFrame({'A': [1, 2], 'B': [5, 3]})
print(df.min(), df.max())  # Min/Max for columns

# 3. Grouping Data
grouped = df.groupby('A').agg(['min', 'max'])
print(grouped)
```

Understanding Means of Values

1. **Arithmetic Mean**
   - Formula: $\bar{x} = \frac{\sum x_i}{n}$
   - Use Case: General average.
   - Example: Average test scores: [80, 85, 90, 75, 70].

     Arithmetic mean=Arithmetic Mean= $\frac{80+85+90+75+70}{5} = 80$

2. **Geometric Mean**
   - Formula: $(\Pi_{i=1}^{n} x_i)^{\frac{1}{n}}$
   - Use Case: Growth rates (e.g., sales growth).
   - Example: Sales growth rates: [1.1, 1.05, 0.9].

   $Geometric\ Mean \approx 1.01$ (1% average growth)

Understanding Means of Values

1. **Harmonic Mean**
   - **Formula**: $H = \frac{n}{\sum \frac{1}{x_i}}$
   - **Use Case**: **Average rates (e.g., speed).**
   - **Example**: **Travel speeds: 60 km/h and 30 km/h.**

   Harmonic mean $= 40\ km/h$
2. **Weighted Mean**
   - **Formula**: $\bar{x}_w = \frac{\sum w_i \cdot x_i}{\sum w_i}$
   - **Use Case**: **Weighted grades or importance.**
   - **Example**: **Grades: 85 (weight 0.7) and 70 (weight 0.3).**

   Weighted Mean $= 80.5$

Python Code Examples for Means


```python
import numpy as np
from scipy.stats import hmean

# Arithmetic Mean
print("Arithmetic Mean:", np.mean([186, 179, 182, 165, 173]))

# Geometric Mean
data = [1.10, 0.88, 1.9, 0.7, 1.25]
print("Geometric Mean:", np.prod(data) ** (1 / len(data)))

# Harmonic Mean
print("Harmonic Mean:", hmean([60, 20]))

# Weighted Mean
print("Weighted Mean:", np.average([90, 70], weights=[0.6, 0.4]))
```

```
Arithmetic Mean: 177.0
Geometric Mean: 1.0998346610476917
Harmonic Mean: 30.0
Weighted Mean: 82.0
```

Median and Mode (Concepts)

1. **Median (Middle Value):**
   - Definition: The central value of sorted data. If the data has an even number of values, it's the average of the two middle values.
   - Use Case: Preferred when data has outliers, as it is less sensitive to extreme values.
   - Example: Heights [160, 165, 170, 175, 180] → Median = 170.
2. **Mode (Most Frequent Value):**
   - Definition: The value that appears most frequently in the data.
   - Use Case: Useful for categorical data or identifying the most common value.
   - Types: Unimodal (1 mode), Bimodal (2 modes), Multimodal (more than 2 modes).
   - Example: Scores [85, 90, 85, 80, 85] → Mode = 85.

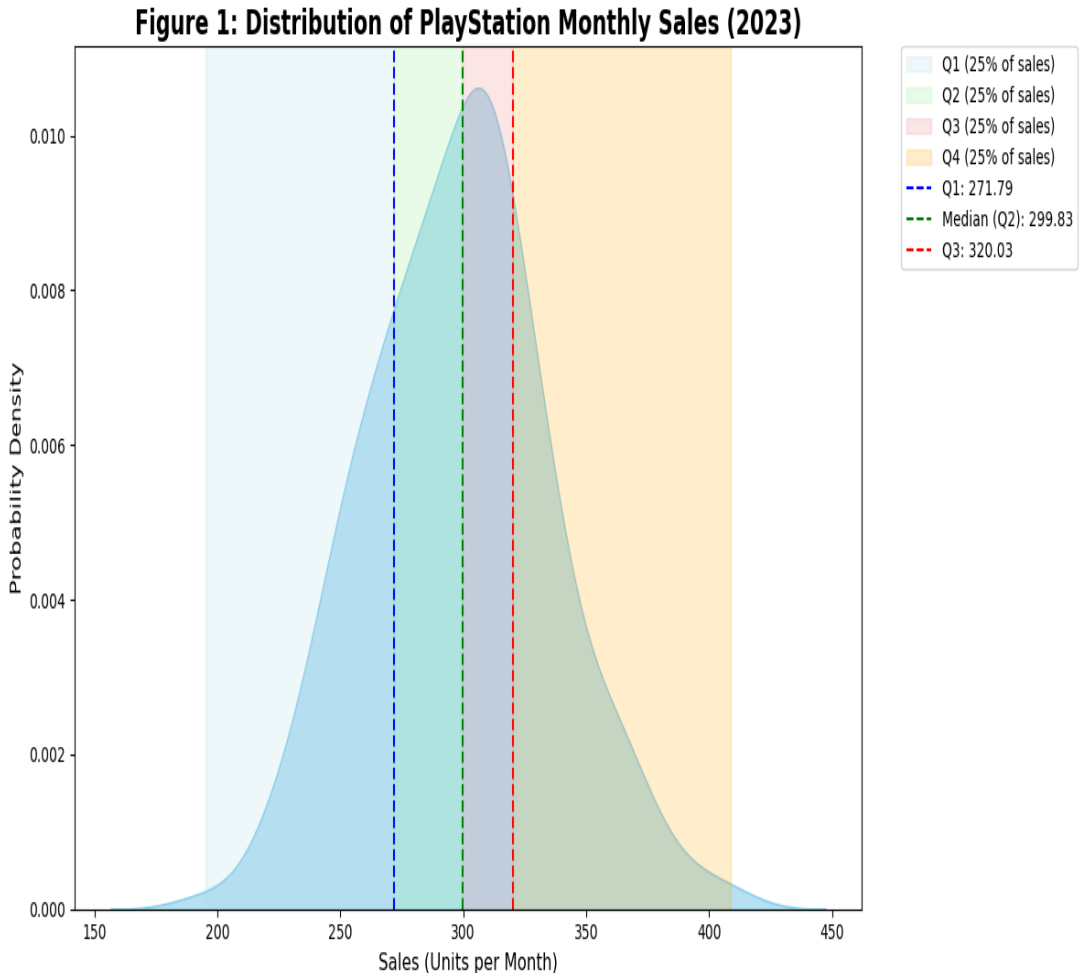
Python Code for Median and Mode

```python
import numpy as np
from scipy import stats
from statistics import multimode

# Sample data
data1 = [160, 165, 170, 175, 180]  # For median
data2 = [85, 90, 85, 80, 85]       # For mode
data3 = [1, 2, 2, 3, 3]            # For multi-mode

# Calculations
median = np.median(data1)
mode_result = stats.mode(data2, keepdims=True).mode.item()  # Compatible with recent versions
multi_mode_result = multimode(data3)

# Output results
print(f"Median: {median}")          # Output: 170
print(f"Mode: {mode_result}")       # Output: 85
print(f"Multi-mode: {multi_mode_result}")  # Output: [2, 3]
```

```
Median: 170.0
Mode: 85
Multi-mode: [2, 3]
```

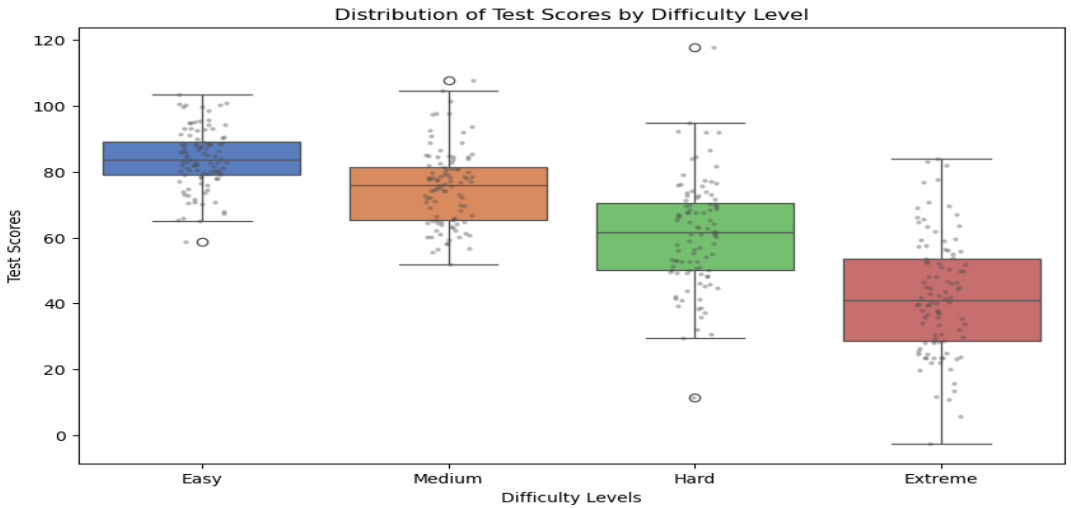
Understanding Quantiles and IQR (Interquartile Range)

- **Quantiles**: Divide data into equal parts (e.g., quartiles, deciles, percentiles).
  - Quartiles: Divide data into 4 equal groups:
    - Q1 (25%): Lower quartile
    - Q2 (50%): Median
    - Q3 (75%): Upper quartile
- **Interquartile Range (IQR)**: The range between Q3 and Q1 (IQR = Q3 - Q1).
  - Helps detect outliers and understand data spread.
- **Why it Matters:**
  - Describes data distribution and skewness.
  - IQR-based thresholds often flag outliers as values below Q1 - 1.5*IQR or above Q3* + 1.5IQR.



Boxplots and Python Implementation

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

# Generate sample data for test scores
np.random.seed(42)
scores_easy = np.random.normal(85, 10, 100)
scores_medium = np.random.normal(75, 12, 100)
scores_hard = np.random.normal(60, 15, 100)
scores_extreme = np.random.normal(40, 20, 100)

scores_data = [scores_easy, scores_medium, scores_hard, scores_extreme]
levels = ['Easy', 'Medium', 'Hard', 'Extreme']

# Create a boxplot with stripplot overlay
plt.figure(figsize=(10, 6))
sns.boxplot(data=scores_data, palette='muted')
sns.stripplot(data=scores_data, jitter=True, size=3, color=".3", alpha=0.4)
plt.xticks(range(len(levels)), levels)
plt.title('Distribution of Test Scores by Difficulty Level')
plt.ylabel('Test Scores')
plt.xlabel('Difficulty Levels')
plt.show()
```



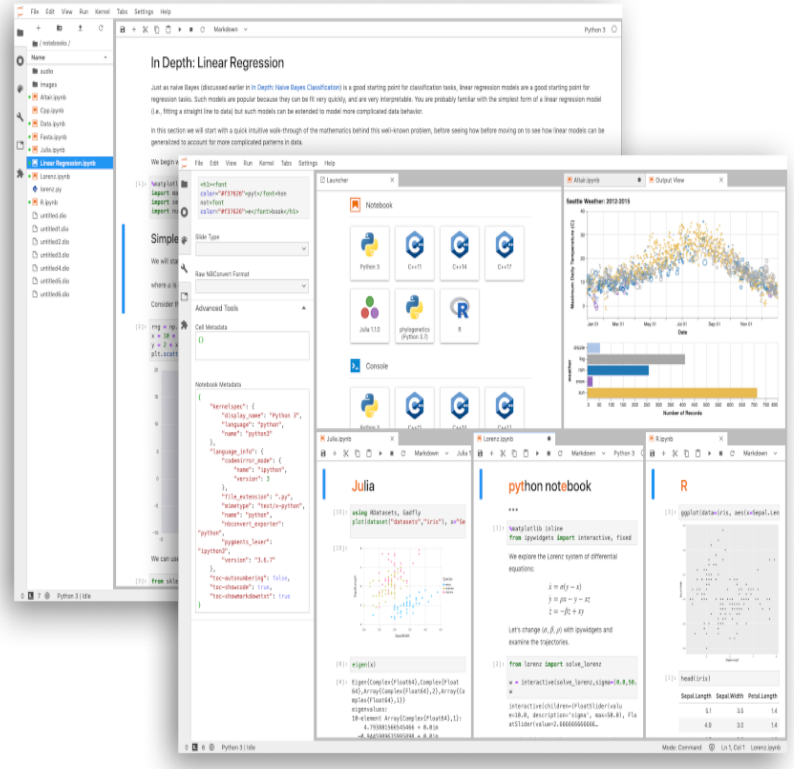
Visual Explanation:

- The shaded box represents values between Q1 and Q3.
- The horizontal line inside the box is the median (Q2).
- "Whiskers" extend to non-outlier data points.
- Points outside whiskers are outliers.

Notebook Review

Walk through how to apply key Python concepts in a Jupyter Notebook:

- Minimum and Maximum Values
- Means of Values (Arithmetic, Geometric, Harmonic, Weighted)
- Median and Mode
- Quantiles and IQR (Interquartile Range)



Kahoot Quiz Time!

Metropolia
University of Applied Sciences

Kahoot!

Let's Test Our Knowledge!

Hands-on Exercise

Form groups (2–3 members).

- Download *Hands-on Exercise #3* from the course page.
- Complete the coding tasks and discuss your solutions.
- Don’t forget to add the names of your group members to the file.
- Submit your completed *Hands-on Exercise* to the course Moodle page or send it to the teacher's email address.

Reference

- Vohra, M., & Patil, B. (2021). A Walk Through the World of Data Analytics. , 19-27. https://doi.org/10.4018/978-1-7998-3053-5.ch002.
- VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. O'Reilly Media. Available at https://jakevdp.github.io/PythonDataScienceHandbook/
- Severance, C. (2016). Python for everybody: Exploring data using Python 3. Charles Severance. Available at https://www.py4e.com/html3/
- McKinney, W. (2017). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter*. O'Reilly Media.