📚 Module 30-31: Spread & Standard Deviation - Complete Notes

🎯 What You'll Learn

Master **range, IQR, variance, and standard deviation** — measuring data variability.

📖 Concept Explained

Measures of Spread

```
Data: [10, 20, 30, 40, 50]

MIN: 10
MAX: 50
RANGE: 50 - 10 = 40

QUARTILES (divide into 4 parts):
Q1 (25%): 20
Q2 (50%): 30 (median)
Q3 (75%): 40

IQR (Interquartile Range): Q3 - Q1 = 20
```

Standard Deviation

```
High std dev = Data is SPREAD OUT
Low std dev = Data is CLUSTERED

Data A: [100, 101, 99, 102, 98]  → Low std dev (consistent)
Data B: [50, 150, 75, 180, 45]  → High std dev (variable)
```

💻 Programming Connection

Code Examples

```python
# Example 1: Range and Basic Spread

def spread_measures(data):
    return {
        "min": min(data),
        "max": max(data),
        "range": max(data) - min(data),
        "count": len(data)
    }

times = [100, 150, 120, 200, 180]
```

```
print(spread_measures(times))
# {'min': 100, 'max': 200, 'range': 100, 'count': 5}
```

```
# Example 2: Quartiles and IQR

def quartiles(data):
    """Calculate Q1, Q2 (median), Q3"""
    sorted_d = sorted(data)
    n = len(sorted_d)

    q2 = sorted_d[n // 2] if n % 2 else (sorted_d[n//2 - 1] + sorted_d[n//2]) / 2

    lower = sorted_d[:n // 2]
    upper = sorted_d[(n + 1) // 2:]

    q1 = sorted(lower)[len(lower) // 2] if lower else None
    q3 = sorted(upper)[len(upper) // 2] if upper else None

    return {"Q1": q1, "Q2": q2, "Q3": q3, "IQR": q3 - q1 if q1 and q3 else None}

data = list(range(1, 21))  # 1 to 20
print(quartiles(data))
```

```
# Example 3: Standard Deviation

from statistics import mean, stdev, pstdev

data = [100, 102, 98, 105, 101]

print(f"Mean: {mean(data)}")
print(f"Sample Std Dev: {stdev(data):.2f}")      # 2.55
print(f"Population Std Dev: {pstdev(data):.2f}")  # 2.28
```

```
# Example 4: Interpreting Standard Deviation

def analyze_consistency(data, label="Data"):
    """Analyze consistency using coefficient of variation"""
    from statistics import mean, stdev

    avg = mean(data)
    std = stdev(data)
    cv = (std / avg) * 100  # Coefficient of variation

    return {
        "label": label,
        "mean": round(avg, 2),
        "std_dev": round(std, 2),
        "cv_percent": round(cv, 2),
        "consistency": "High" if cv < 10 else "Medium" if cv < 25 else "Low"
```

```
    }

consistent = [100, 102, 98, 101, 99]
variable = [50, 150, 75, 200, 100]

print(analyze_consistency(consistent, "Consistent"))
# cv: ~1.7%, consistency: High

print(analyze_consistency(variable, "Variable"))
# cv: ~50%, consistency: Low
```

```
# Example 5: Outlier Detection using IQR

def find_outliers_iqr(data):
    """Detect outliers using IQR method"""
    sorted_d = sorted(data)
    n = len(sorted_d)

    q1 = sorted_d[n // 4]
    q3 = sorted_d[3 * n // 4]
    iqr = q3 - q1

    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    outliers = [x for x in data if x < lower_bound or x > upper_bound]

    return {
        "outliers": outliers,
        "bounds": (lower_bound, upper_bound),
        "count": len(outliers)
    }

data = [10, 12, 14, 15, 18, 22, 24, 100]
print(find_outliers_iqr(data))
# {'outliers': [100], 'bounds': (-1.5, 40.5), 'count': 1}
```

🧪 SDET/Testing Application

```
# SDET Scenario: Performance Stability Analysis

def check_performance_stability(response_times, max_cv=15):
    """Check if performance is stable (low variation)"""
    analysis = analyze_consistency(response_times)

    return {
        **analysis,
        "stable": analysis["cv_percent"] < max_cv,
        "verdict": "✅ Stable" if analysis["cv_percent"] < max_cv else "⚠️ Variable"
```

```
    }

times = [150, 152, 148, 155, 147]  # Consistent
print(check_performance_stability(times))
# ✅ Stable (low CV)

times = [100, 300, 150, 500, 200]  # Variable
print(check_performance_stability(times))
# ⚠️ Variable (high CV)
```

🔑 Key Takeaways

✅ **Range = Max - Min**
✅ **IQR = Q3 - Q1** — Robust to outliers
✅ **Low Std Dev = Consistent**
✅ **High Std Dev = Variable**
✅ **CV (Coefficient of Variation)** — Compare variability across datasets

---

💾 *Save as:* `Module_30_31_Spread_StdDev.md`