📚 Module 28-29: Data Collection & Central Tendency - Complete Notes

🎯 What You'll Learn

Master **data organization** and **mean, median, mode** — the foundation of all data analysis.

📖 Concept Explained

Data Types

```
CATEGORICAL: Labels/Names (pass, fail, browser)
NUMERICAL: Measurable (response time, count)
ORDINAL: Ordered categories (High, Medium, Low)
```

Central Tendency: Finding the "Typical" Value

```
Data: [10, 20, 20, 30, 100]

MEAN (Average): Sum / Count
    (10+20+20+30+100) / 5 = 36
    ⚠️ Affected by outliers (100 pulls it up!)

MEDIAN (Middle): Sort, pick middle
    [10, 20, 20, 30, 100] → 20
    ✅ Robust to outliers

MODE (Most frequent): What appears most
    20 appears twice → Mode = 20
    ✅ Works for categorical data
```

When to Use Each

```
USE MEAN when: Data is symmetric, no outliers
USE MEDIAN when: Data is skewed, has outliers
USE MODE when: Categorical data, most common value
```

💻 Programming Connection

Code Examples

```python
# Example 1: Mean (Average)

def mean(data):
    if not data:
        return None
    return sum(data) / len(data)

response_times = [100, 150, 120, 200, 130]
print(f"Mean: {mean(response_times)}")  # 140.0

# Built-in
from statistics import mean as stat_mean
print(stat_mean(response_times))  # 140.0
```

```
# Example 2: Median (Middle Value)

def median(data):
    if not data:
        return None
    sorted_data = sorted(data)
    n = len(sorted_data)
    mid = n // 2

    if n % 2 == 0:
        return (sorted_data[mid - 1] + sorted_data[mid]) / 2
    else:
        return sorted_data[mid]

print(median([1, 3, 5, 7, 9]))     # 5 (middle of 5)
print(median([1, 3, 5, 7]))       # 4.0 (average of 3 and 5)
```

```
# Example 3: Mode (Most Frequent)

from collections import Counter

def mode(data):
    if not data:
        return None
    counts = Counter(data)
    max_count = max(counts.values())
    modes = [k for k, v in counts.items() if v == max_count]
    return modes[0] if len(modes) == 1 else modes

print(mode([1, 2, 2, 3, 3, 3]))  # 3
print(mode(["pass", "fail", "pass", "pass"]))  # "pass"
```

```
# Example 4: Mean vs Median - The Outlier Effect

times_normal = [100, 102, 98, 105, 101]
times_outlier = [100, 102, 98, 105, 1000]

print("Normal data:")
print(f"  Mean: {mean(times_normal):.1f}")    # 101.2
print(f"  Median: {median(times_normal)}")   # 101
print("With outlier:")
print(f"  Mean: {mean(times_outlier):.1f}")   # 281.0 ← Skewed!
print(f"  Median: {median(times_outlier)}")  # 102 ← Robust!
```

```
# Example 5: Frequency Table

def frequency_table(data):
    """Count occurrences of each value"""
    return dict(Counter(data))

statuses = ["pass", "pass", "fail", "pass", "skip"]
print(frequency_table(statuses))
# {'pass': 3, 'fail': 1, 'skip': 1}
```

🧪 SDET/Testing Application

```
# SDET Scenario: Analyze Test Results

def analyze_results(test_results):
    """Comprehensive test result analysis"""
    durations = [r['duration'] for r in test_results]
    statuses = [r['status'] for r in test_results]

    return {
        "total_tests": len(test_results),
        "duration_mean": round(mean(durations), 2),
        "duration_median": median(durations),
        "most_common_status": mode(statuses),
        "status_counts": frequency_table(statuses)
    }

results = [
    {"test": "login", "status": "pass", "duration": 120},
    {"test": "search", "status": "pass", "duration": 350},
    {"test": "checkout", "status": "fail", "duration": 500},
    {"test": "cart", "status": "pass", "duration": 180},
]

print(analyze_results(results))
```

```
# SDET Scenario: When to Use What

def recommend_central_measure(data):
    """Recommend best measure based on data"""
    avg = mean(data)
    med = median(data)
    # If mean and median differ significantly, data is skewed
    diff_pct = abs(avg - med) / med * 100 if med else 0

    if diff_pct > 20:
        return {
            "recommendation": "Use MEDIAN",
            "reason": f"Data appears skewed (mean={avg:.1f}, median={med})",
            "mean": avg,
            "median": med
        }
    else:
        return {
            "recommendation": "Use MEAN",
            "reason": "Data is relatively symmetric",
            "mean": avg,
            "median": med
        }

# Normal data
print(recommend_central_measure([100, 102, 98, 105, 101]))
# Use MEAN

# Skewed data
print(recommend_central_measure([100, 102, 98, 105, 1000]))
# Use MEDIAN
```

🔑 Key Takeaways

✅ **Mean = Sum/Count** — Affected by outliers
✅ **Median = Middle** — Robust to outliers
✅ **Mode = Most frequent** — For categories
✅ **Skewed data → Use median**
✅ **Symmetric data → Mean is fine**

💾 *Save as:* `Module_28_29_Central_Tendency.md`