Statistics with Lists

Calculating Central Tendency in Python

Review: What We Know

Statistics

- **Mean**: The average of all values
- **Median**: The middle value when sorted
- **Mode**: The most frequent value
- **Standard Deviation**: How spread out the data is

Lists

- Storing multiple values: `scores = [85, 92, 78]`
- Accessing items: `scores[0]`
- Adding items: `scores.append(95)`
- Iterating: `for score in scores:`

Calculating the Mean

The mean is the sum of all values divided by the count:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

```python
def get_mean(values):
    total = 0
    for value in values:
        total = total + value
    count = len(values)
    return total / count

scores = [85, 92, 78, 88, 95, 73]

# Calculate mean
mean_score = get_mean(scores)
print(f"Mean Score: {mean_score:.2f}")
```

```
Mean Score: 85.17
```

Finding the Median

The median is the middle value when sorted:

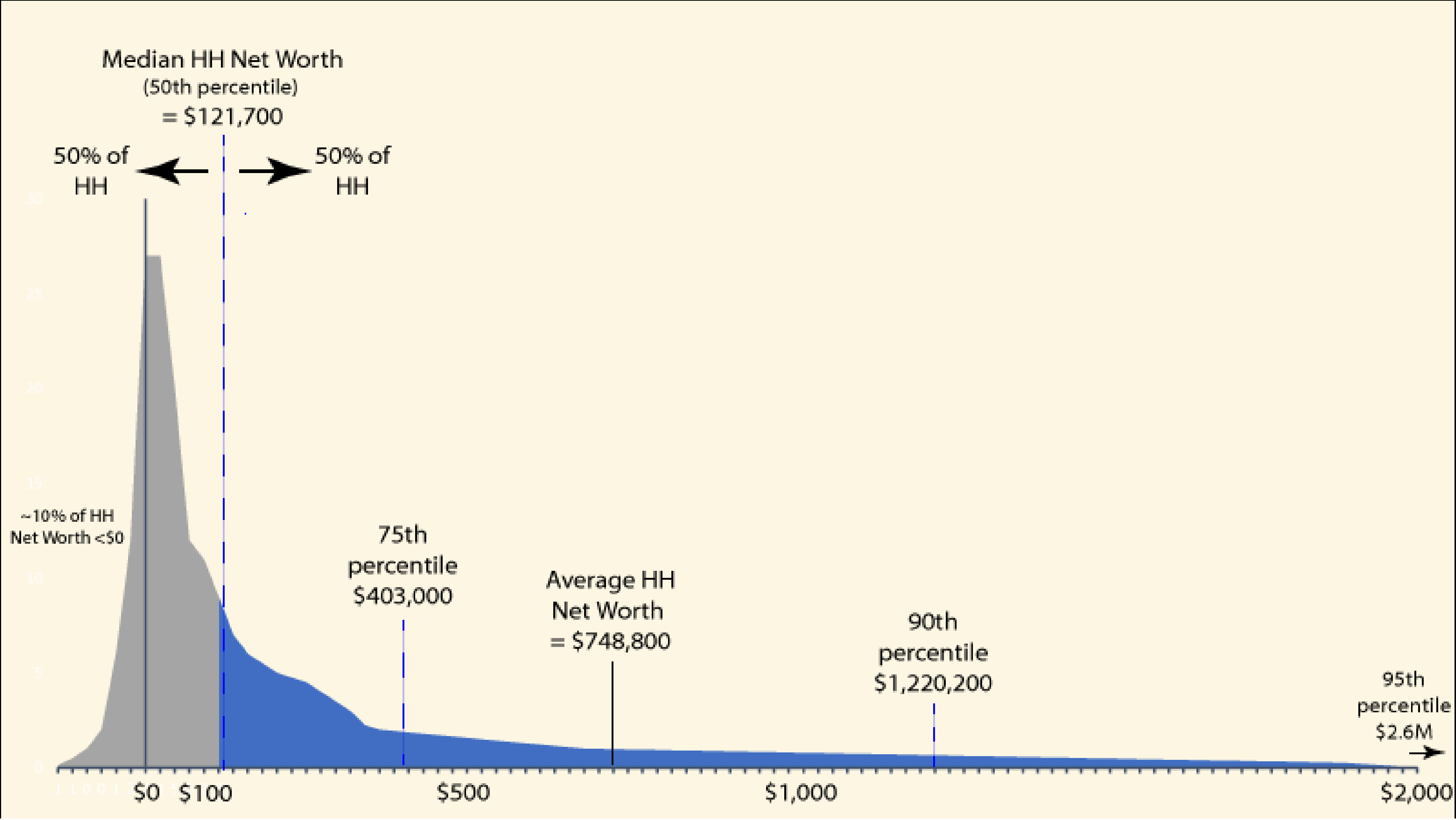
```python
def get_median(values):
    sorted_values = sorted(values)
    n = len(sorted_values)
    if n % 2 == 0:  # Even number of values
        mid1 = sorted_values[n // 2 - 1]
        mid2 = sorted_values[n // 2]
        return (mid1 + mid2) / 2
    else:  # Odd number of values
        return sorted_values[n // 2]

values = [23, 45, 12, 67, 34, 89, 56]
median = get_median(values)
print(f"Median: {median}")
```

```
Median: 45
```

Why Median Matters: U.S. Net Worth by Age

Age of Head of Family	Median Net Worth	Average Net Worth
Under 35	$39,040	$183,380
35-44	$135,300	$548,070
45-54	$246,700	$971,270
55-64	$364,270	$1,564,070
65-74	$410,000	$1,780,720
75+	$334,700	$1,620,100

https://www.nerdwallet.com/article/finance/average-net-worth-by-age



Median HH Net Worth
(50th percentile)
= $121,700
50% of
HH
50% of
HH
~10% of HH
Net Worth <$0
75th
percentile
$403,000
Average HH
Net Worth
= $748,800
90th
percentile
$1,220,200
95th
percentile
$2.6M
$0
$100
$500
$1,000
$2,000

Understanding Skew

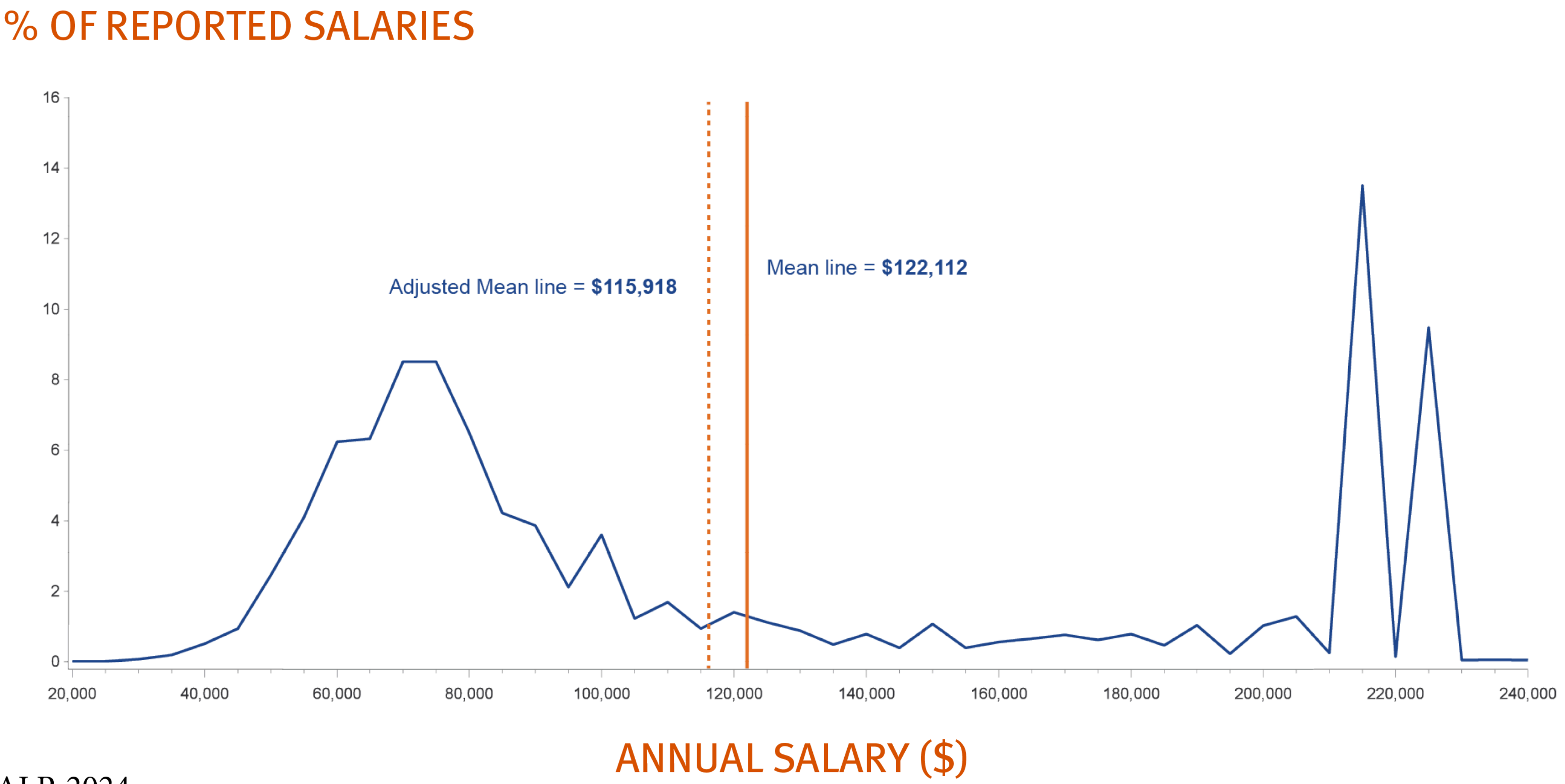
When data is **skewed**, mean and median tell different stories:

```
# Right-skewed data (long tail on right)
salaries = [30, 35, 40, 45, 50, 55, 60, 70, 250]

salaries_sum = 0
for salary in salaries:
    salaries_sum += salary

mean_sal = salaries_sum / len(salaries)
median_sal = sorted(salaries)[len(salaries)//2]

print("Right-skewed (income-like):")
print(f"Mean: ${mean_sal:.0f}k")
print(f"Median: ${median_sal:.0f}k")

if mean_sal > median_sal:
    print("Mean > Median = Right skew!")
    print("A few high values pull the mean up")
```

```
Right-skewed (income-like):
Mean: $71k
Median: $50k
```

Rule of thumb:

- Mean > Median → Right skew
  - Example: income, wealth
- Mean < Median → Left skew
  - Example: age at death
- Mean ≈ Median → Symmetric
  - Example: height, weight


% OF REPORTED SALARIES
16
14
12
10
8
6
4
2
0
Adjusted Mean line = $115,918
Mean line = $122,112
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
220,000
240,000
ANNUAL SALARY ($)

What is a Histogram?

A histogram is a chart that shows how frequently different values appear in your data:

Why use histograms?

- See patterns at a glance (symmetric? skewed? bimodal?)
- Spot outliers immediately
- Understand spread better than just mean/median
- Make better decisions based on distribution shape

```python
scores = [72, 85, 90, 78, 92, 88, 75, 95, 82, 88, 91, 79, 8

# Count scores in ranges (bins)
range_60_70 = 0
range_70_80 = 4   # 72, 75, 78, 79
range_80_90 = 7   # 82, 83, 85, 86, 88, 88, 90
range_90_100 = 4  # 90, 91, 92, 95

def make_bar(count):
    bar = ""
    for i in range(count):
        bar += "█"
    return bar

print("60-70 : " + make_bar(range_60_70))
print("70-80 : " + make_bar(range_70_80))
print("80-90 : " + make_bar(range_80_90))
print("90-100: " + make_bar(range_90_100))
```

```
60-70 :
70-80 : ████
80-90 : ███████
90-100: ████
```

Calculating Standard Deviation

How spread out is our data?

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

```python
values = [2, 4, 4, 4, 5, 5, 7, 9]

# Step 1: Calculate mean
sum_values = 0
for x in values:
    sum_values += x
mean = sum_values / len(values)
print(f"Mean: {mean}")

# Step 2: Calculate squared differences
sum_squared_diffs = 0
for x in values:
    diff = x - mean
    sum_squared_diffs += diff ** 2

# Step 3: Calculate variance and std dev
variance = sum_squared_diffs / (len(values) - 1)
std_dev = variance ** 0.5

print(f"Standard deviation: {std_dev:.2f}")
```

```
Mean: 5.0
Standard deviation: 2.14
```

Comparing Spread

Standard deviation helps us understand consistency:

```python
# Two classes with same mean
class_a = [82, 84, 85, 85, 86, 88]
class_b = [70, 75, 85, 85, 95, 100]

mean_a = sum(class_a) / len(class_a)
mean_b = sum(class_b) / len(class_b)

# Calculate std dev for both
def std_dev(data):
    mean = sum(data) / len(data)
    squared_diffs = [(x - mean)**2 for x in data]
    return (sum(squared_diffs) / (len(data) - 1)) ** 0.5

print(f"Class A - Mean: {mean_a:.1f}, Std Dev: {std_dev(class_a):.1f}")
print(f"Class B - Mean: {mean_b:.1f}, Std Dev: {std_dev(class_b):.1f}")
print("\nClass B has more variation in scores!")
```

```
Class A - Mean: 85.0, Std Dev: 2.0
Class B - Mean: 85.0, Std Dev: 11.4

Class B has more variation in scores!
```

Quartiles

Understanding data distribution:

```python
scores = [65, 70, 72, 75, 78, 80, 82, 85, 88, 90, 92, 95, 98]
sorted_scores = sorted(scores)

# Quartiles
n = len(sorted_scores)
q1_index = n // 4
q2_index = n // 2  # median
q3_index = 3 * n // 4

print(f"Q1 (25th percentile): {sorted_scores[q1_index]}")
print(f"Q2 (50th percentile/median): {sorted_scores[q2_index]}")
print(f"Q3 (75th percentile): {sorted_scores[q3_index]}")
```

```
Q1 (25th percentile): 75
Q2 (50th percentile/median): 82
Q3 (75th percentile): 90
```

Percentiles

Percentiles indicate the relative standing of a value within a dataset.

```python
scores = [65, 70, 72, 75, 78, 80, 82, 85, 88, 90, 92, 95, 98]
sorted_scores = sorted(scores)
n = len(sorted_scores)

# What percentile is a score of 85?
score = 85

num_below_score = 0
for s in sorted_scores:
    if s < score:
        num_below_score += 1

percentile = (num_below_score / n) * 100
print(f"\nA score of {score} is at the {percentile:.0f}th percentile")
```

```
A score of 85 is at the 54th percentile
```

When to Use Each Measure

Mean

- Good for symmetric data
- Sensitive to outliers
- Use for: test scores, temperatures

Median

- Robust to outliers
- Better for skewed data
- Use for: income, home prices

Standard Deviation

- Measures consistency
- Higher = more spread
- Use for: quality control, risk assessment

Quartiles/Percentiles

- Understand distribution shape
- Identify outliers
- Compare relative standing

Exercise: Housing Prices

bigd103.link/housing-prices