Using Statistics with Jupyter Notebooks

Introduction to Statistical Averaging

Statistics *(noun)* - a branch of mathematics dealing with the *collection*, *analysis*, *interpreation*, and *presentation* of masses of numerical data.

Allowoing for one to make **informned** decisions as a result.

Averages

Mean

Mean - summatizes an entire data set with a **single number** representing the data's *center point*.

Proof

$$mean = \frac{sum\ of\ the\ terms}{number\ of\ terms}$$

$$mean = \frac{\sum n}{n}$$

Example Data: $[88, 5, 4, 16, 72, 22, 75, 74, 27, 9]$

Count: $[10]$

Mean: $[39.2]$

$$mean = \frac{(88 + 5 + 4 + 16 + 72 + 22 + 75 + 74 + 27 + 9)}{10}$$
$$mean = 39.2$$

```
my_nums = [88, 5, 4, 16, 72, 22, 75, 74, 27, 9]
```

```
count = len(my_nums)
mean = sum(my_nums)/count
print(f"Mean: {mean}")
```

```
Mean: 39.2
```

```
import numpy as np
```

```
# Mean - numpy
mean = np.mean(my_nums) # sum of total / count
print(f"Mean: {mean}")
```

```
Mean: 39.2
```

Median

Median - The middle value of an ordered data set.

Proof Odd:

$$\frac{n+1}{2}^{th} term$$

Even:

$$\frac{\frac{n}{2}^{th} term + (\frac{n}{2}+1)^{th} term}{2}$$

Example Data: $[4, 5, 9, 16, 22, 27, 72, 74, 75, 88]$

Find middle value(s). $[4, 5, 9, 16, \mathbf{22}, \mathbf{27}, 72, 74, 75, 88]$

$$median = \frac{22+27}{2} = 24.5$$

```
sorted_my_nums = sorted(my_nums)
print(sorted_my_nums)
```

```
[4, 5, 9, 16, 22, 27, 72, 74, 75, 88]
```

```
median = None
if (count % 2 == 0):
    median = (sorted_my_nums[count // 2] + sorted_my_nums[count // 2-1])/2
else:
    median = median = sorted_my_nums[count // 2]
print(f"Median: {median}")
```

```
Median: 24.5
```

```
# Median - numpy
median = np.median(sorted_my_nums) # midle value from data
print(f"Median: {median}")
```

```
Median: 24.5
```

Mode

Mode - is/are the value(s) repeated most often in a data set.

- The value with the highest frequency.

Example Data: $[2, 4, 5, 5, 7, 8, 8, 9, 10]$

Value	2	4	5	7	8	9	10
Frequency	1	1	2	1	2	1	1

$$Mode = [5, 8]$$

```
# Version 1
my_nums = [2, 4, 5, 5, 7, 8, 8, 9, 10]
my_dict = {num: my_nums.count(num) for num in my_nums}
mode = [k for k, v in my_dict.items() if v == max(my_dict.values())]
#mode = [k for num in my_nums if my_nums.count(num) == ]
print(mode)
```

```
[5, 8]
```

```
import scipy
```

```
# Version 2
my_dict = {}
for num in my_nums:
    my_dict[num] = my_nums.count(num)
mode = []
for k, v in my_dict.items():
    if v == max(my_dict.values()):
        mode.append(k)
print(mode)
```

```
[5, 8]
```

```
# Version 3
# Mode - scipy
mode = scipy.stats.mode(my_nums) # value(s) with the highest frequency
print(f"Mode: {mode}")
```

```
Mode: ModeResult(mode=np.int64(5), count=np.int64(2))
```

Spread of Data Practice

Domain

Domain - signifies the entire range of values that data points can assume.

```
my_nums = np.random.randint(15_000, 200_000, 40)  # 1d array with 5 evenly spaced numbers be
print(my_nums)
max_value = np.max(my_nums)
min_value = np.min(my_nums)
print(f'Max: ${max_value}, Min: ${min_value}')
domain = {'min': min_value, 'max': max_value}
print(f'Domain: ${domain.get("min")} -> ${domain.get("max")}')
```

```
[163516  92536  35828 113115  40328 177618  59970  55754 181187  84133
  33389  46045 195235  35201  18001  95988  23600 174417 168787  80128
  55725  47884 175278 101028 108563  52632 179003 104591  19843 111091
```

```
  58608  37739  39786 187091 160324 110610  42424 139812  33655 186660]
Max: $195235, Min: $18001
Domain: $18001 -> $195235
```

Range

Range - the spread between the maximum and minimum values.

$$range = (n - n_1)$$

$$range = max - min$$

$$n_1 = \$22868, n = 197281$$

$$range = \$174413$$

```
range  = max_value - min_value
print(f'Range: ${range}')
```

```
Range: $177234
```

```
rng = int(max_value) - int(min_value)
print(f'Range: ${rng}')
```

```
Range: $177234
```

Higher Concepts

Variance

Variance - measures how far each value in the data set is from the mean. It is the average of the squared differences from the mean.

$$Variance = \frac{\sum(x_i - \mu)^2}{n - 1}$$

Where x_i is each value, μ is the mean, and n is the number of values.

Variance (Calculated with variables)

Compute sample variance step-by-step using variables: mean, deviations, squared deviations, sum, and divide by (n-1).

```
# Manual sample variance calculation using variables
n = len(my_nums)
mean = sum(my_nums) / n
deviations = [x - mean for x in my_nums]
squared_deviations = [d ** 2 for d in deviations]
sum_sq_dev = sum(squared_deviations)
# sample variance uses n-1
```

```
sample_variance = sum_sq_dev / (n - 1) if n > 1 else 0
population_variance = sum_sq_dev / n

print(f"n = {n}")
print(f"Mean = {mean}")
# print(f"Deviations = {deviations}")
# print(f"Squared deviations = {squared_deviations}")
print(f"Sum of squared deviations = {sum_sq_dev}")
print(f"Sample variance (n-1): {sample_variance}")
print(f"Population variance (n): {population_variance}")
```

```
n = 40
Mean = 89981.125
Sum of squared deviations = 136262172230.375
Sample variance (n-1): 3493901852.0608974
Population variance (n): 3406554305.759375
```

```
print(f"Numpy sample variance (ddof=1): {np.var(my_nums, ddof=1)}")
print(f"Numpy population variance (ddof=0): {np.var(my_nums, ddof=0)}")
```

```
Numpy sample variance (ddof=1): 3443971838.9429483
Numpy population variance (ddof=0): 3357872542.9693747
```

Standard Deviation

Standard Deviation - measures the amount of variation or dispersion in a set of values. It is the square root of the variance.

$$Standard\ Deviation = \sqrt{Variance}$$

A low standard deviation means values are close to the mean, while a high standard deviation means values are spread out.

Standard Deviation (Calculated with variables)

Compute standard deviation step-by-step using the variance: take the square root of the sample (n-1) and population (n) variances and compare to NumPy's results.

```
# Manual standard deviation calculation and comparison to NumPy
import math
# standard deviations
sample_std = math.sqrt(sample_variance)
population_std = math.sqrt(population_variance)

print(f"n = {n}")
print(f"Mean = {mean}")
print(f"Sample variance (n-1) = {sample_variance}")
```

```python
print(f"Sample standard deviation (sqrt of n-1 variance) = {sample_std}")
print(f"Population variance (n) = {population_variance}")
print(f"Population standard deviation (sqrt of n variance) = {population_std}")
```

```
n = 40
Mean = 95678.075
Sample variance (n-1) = 3443971838.942949
Sample standard deviation (sqrt of n-1 variance) = 58685.363072430155
Population variance (n) = 3357872542.969375
Population standard deviation (sqrt of n variance) = 57947.153018671896
```

```python
np_sample_std = np.std(my_nums, ddof=1)
np_population_std = np.std(my_nums, ddof=0)
print(f"NumPy sample std (ddof=1) = {np_sample_std}")
print(f"NumPy population std (ddof=0) = {np_population_std}")
```

```
NumPy sample std (ddof=1) = 58685.36307243015
NumPy population std (ddof=0) = 57947.15301867189
```

Practice Prompt

Try generating a new random dataset and calculate its mean, variance, and standard deviation. Visualize the results with a histogram.