NNSE 784
Advanced Analytics Methods

Instructor: F Doyle (CESTM L210)

MW 4:30 – 5:50, NFN 203

Slide Set #3
Descriptive Statistics

Outline for lecture

- First, we will look at key concepts of descriptive statistics using the existing salary data set.

- Secondly, we will apply these concepts to begin examining one feature of the Pima Indian Diabetes Data Set

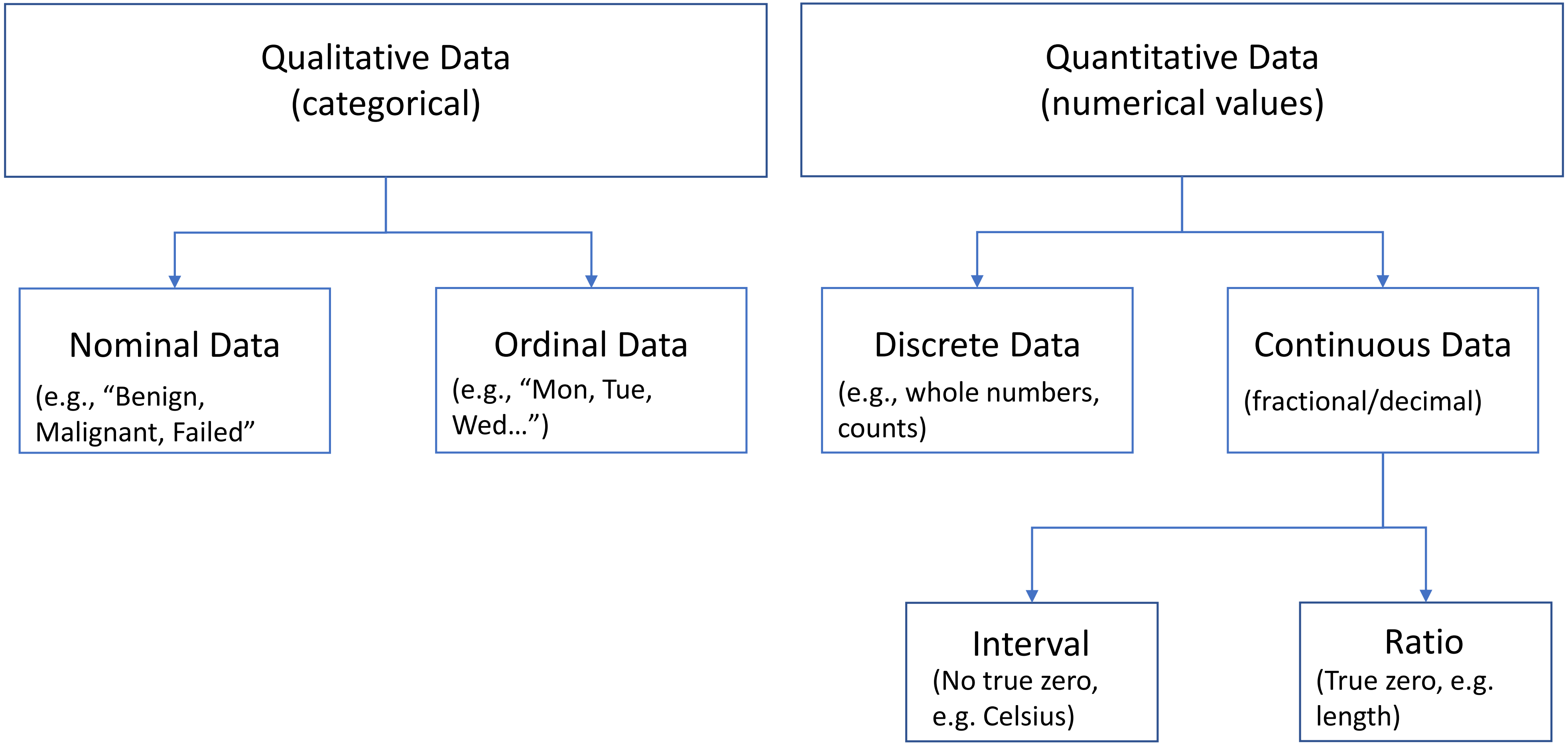
Descriptive Statistics

- Describes the characteristics of a given dataset
- Allows you to organize and summarize a potentially large amount of data and interpret for meaningful analysis
- Does not necessarily aim to reach a conclusion or test a hypothesis
- Can be applied to a full population or a subset thereof (i.e., "sample")
- Graphical plots are often used in conjunction with quantitative measures to increase clarity
- Three major categories
  1. **Frequency Distribution** – how often do particular values occur (histograms, pie charts, etc.)
  2. **Central Tendency** – what value is most representative (mean, median, mode)
  3. **Variability/Dispersion** – how are values distributed/spread (range, variance, standard deviation)

Statistical Notation Reference

$\sum$	Summation	X	An individual value, an observation
s	The standard deviation of sample data	X_1	A particular (1st) individual value
σ	The standard deviation of population data	X_i	For each, all, individual values
s^2	The variance of sample data	$\overline{X}$	The mean, average of sample data
σ^2	The variance of population data	$\overline{\overline{X}}$	The grand mean, grand average
R	The range of data	μ	The mean of population data
$\overline{R}$	The average range of data	p	A proportion of sample data
k	Multi-purpose notation, i.e. # of subgroups, # of classes	P	A proportion of population data
$\lvert y \rvert$	The absolute value of some term	n	Sample size
$>, <$	Greater than, less than	N	Population size
$\geq, \leq$	Greater than or equal to, less than or equal to		

Common Types of Data



Ordered Values

- Data for any given variable, as collected, usually isn’t found in order of value
- Many of the calculations that are performed in descriptive statistics require that numeric values for a variable be in order, as they use proportional positions (e.g., quartiles) to convey characteristics of the data.
- Most of the tools we will use do this for you, but we need to understand why it is important and how we could do it ourselves if needed
- One of the key things that ordering does is allows us to “bin” (group) observations to produce a frequency distribution (histogram)

Frequency Distribution (Histogram)

- Each vertical bar is a "bin" whose width spans a range of values on the x axis.
- The height of each bar corresponds to the number of observations of the variable whose value falls within its range
- The number of intervals (bin widths) chosen is important as to few results in a loss of information and too many results in a lack of summary
- The seaborn histplot() function [distplot() in older distributions] uses an internal, default logic to decide how many intervals to use. This can be overridden (e.g., bins=10).
- Example guideline (don't memorize!)

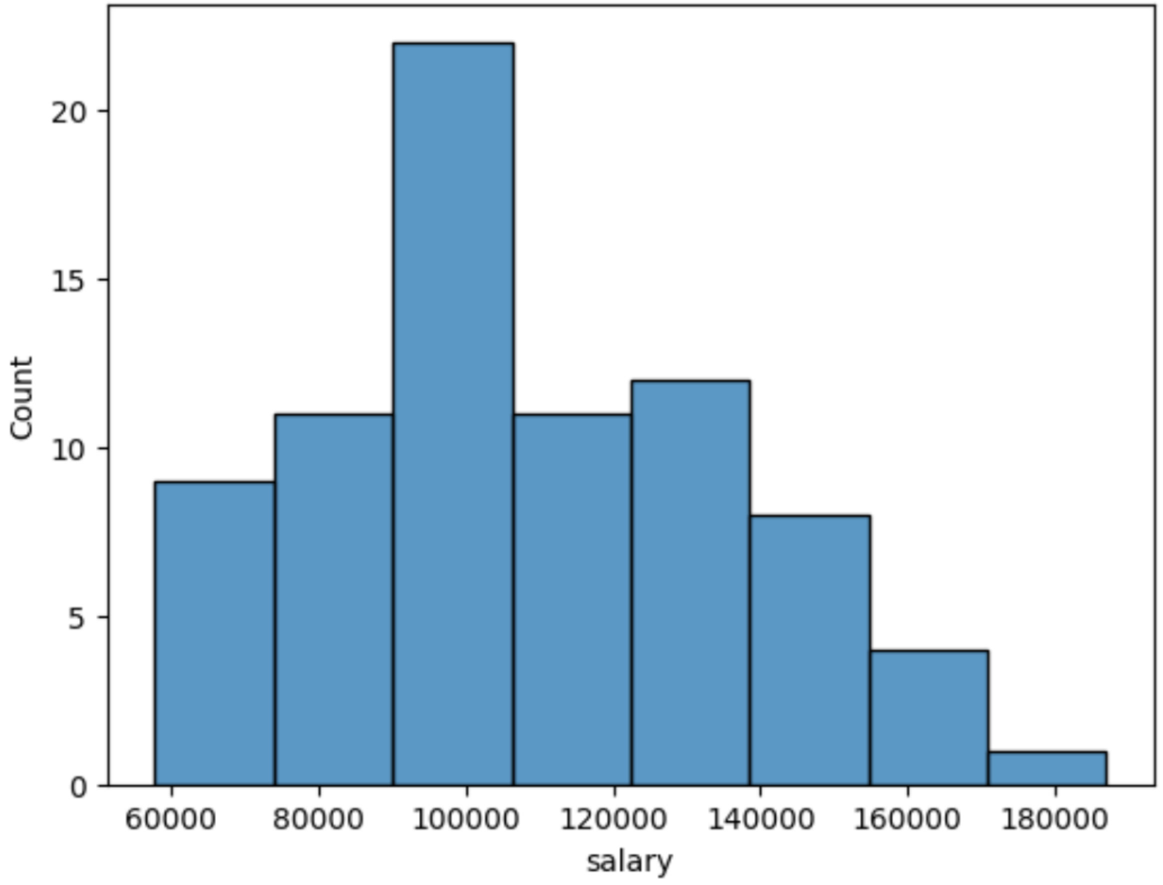
$$k = 1 + 3.322(\log_{10} n) \quad \Leftarrow \textit{Sturges's rule}$$

$$w = \frac{R}{k}$$

w = bin width
R = range between smallest and largest value

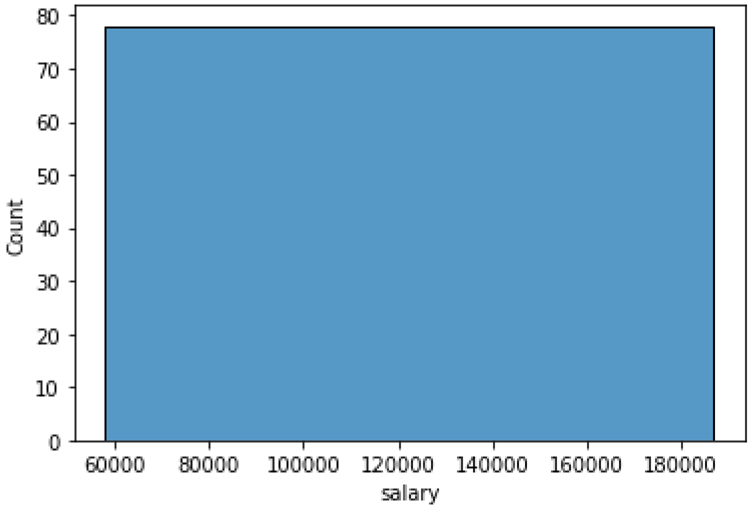
```python
import pandas as pd
import seaborn as sns
df = pd.read_csv("./Salaries.csv")
sns.histplot(df['salary'], kde=False)
```

```
<AxesSubplot:xlabel='salary', ylabel='Count'>
```



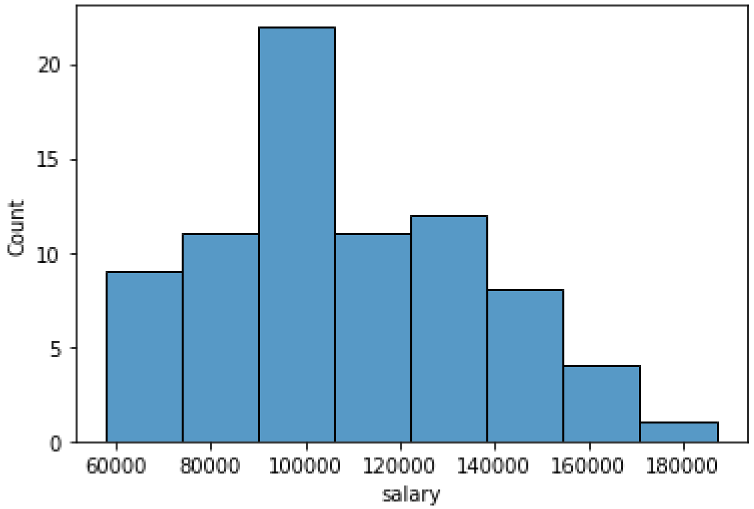
Why Group Size ("Bin Width") Matters

1 Big Bin



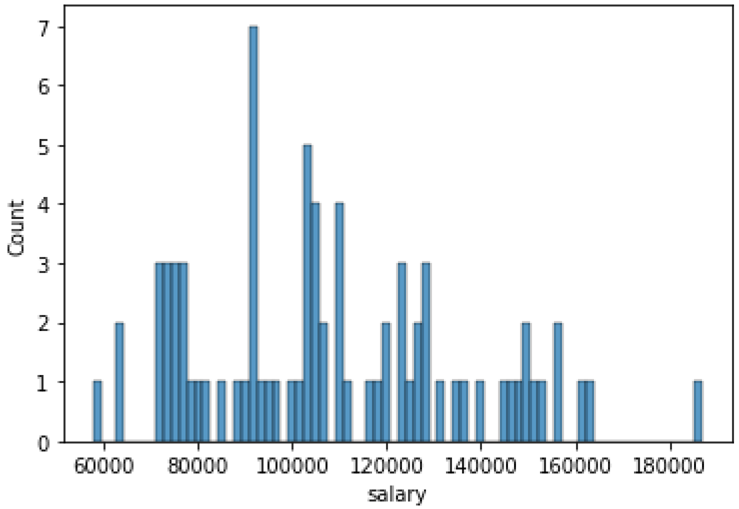
Too few group intervals (bins) = loss of information

8 Bins



Appropriate number of group intervals (bins) = good summary presentation

78 Small Bins

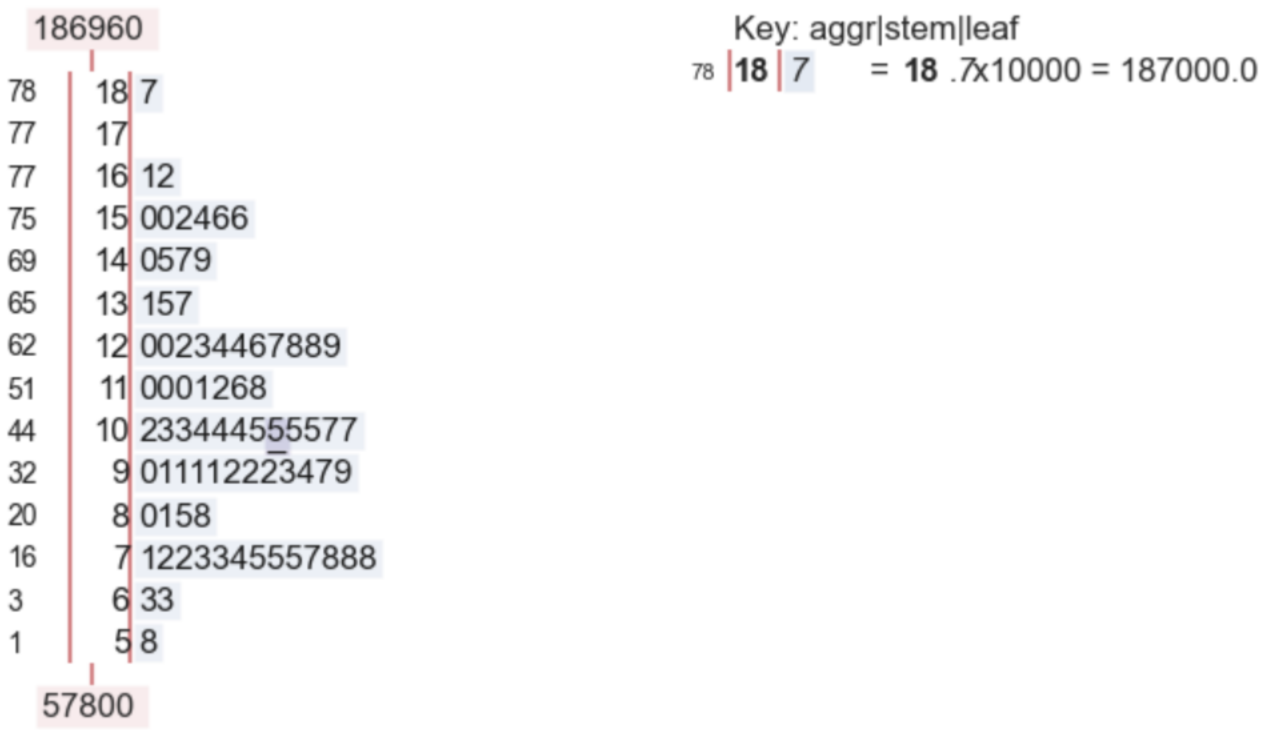


Too many group intervals (bins) = no summarization benefit

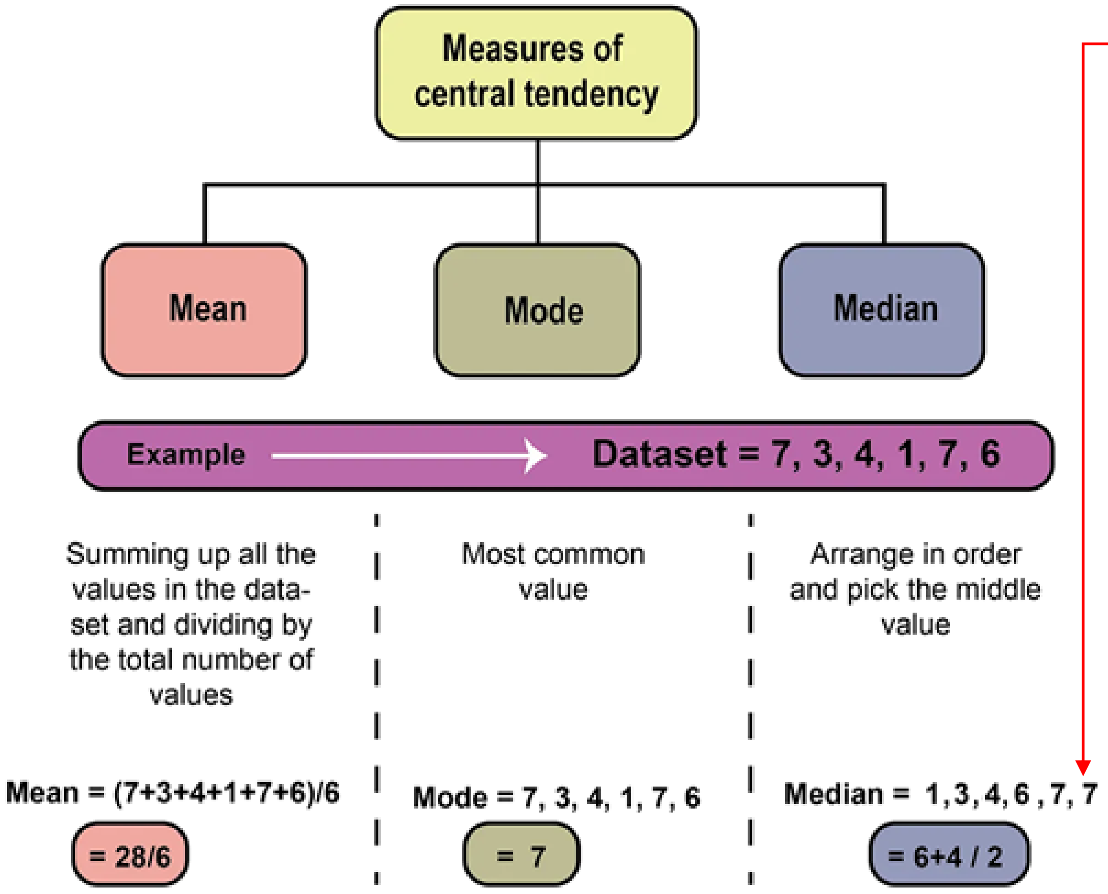
Tangent - Stem and Leaf Display

- Another way of representing this type of data, not generally seen in recent years.
- We will not be using it, but it is worth considering how many ways data can be represented.

```python
import stemgraphic
fig, ax = stemgraphic.stem_graphic(df['salary'])
```



Central Tendency - Mean, Median and Mode



Median, aka "50th percentile" (or quartile 2 "Q2"). Calculating percentiles require values to be sorted.

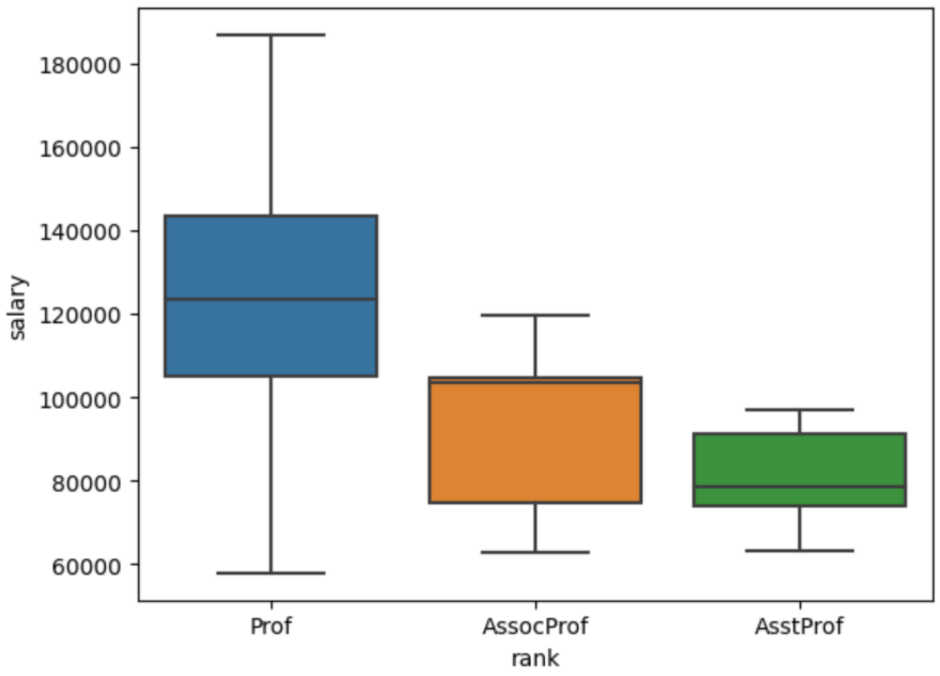
Quartiles are special cases of *location parameters* that correspond to points on the x axis of a histogram

Q1 = ¼ (n + 1)th term
Q2 = ½ (n + 1)th term
Q3 = ¾ (n + 1)th term

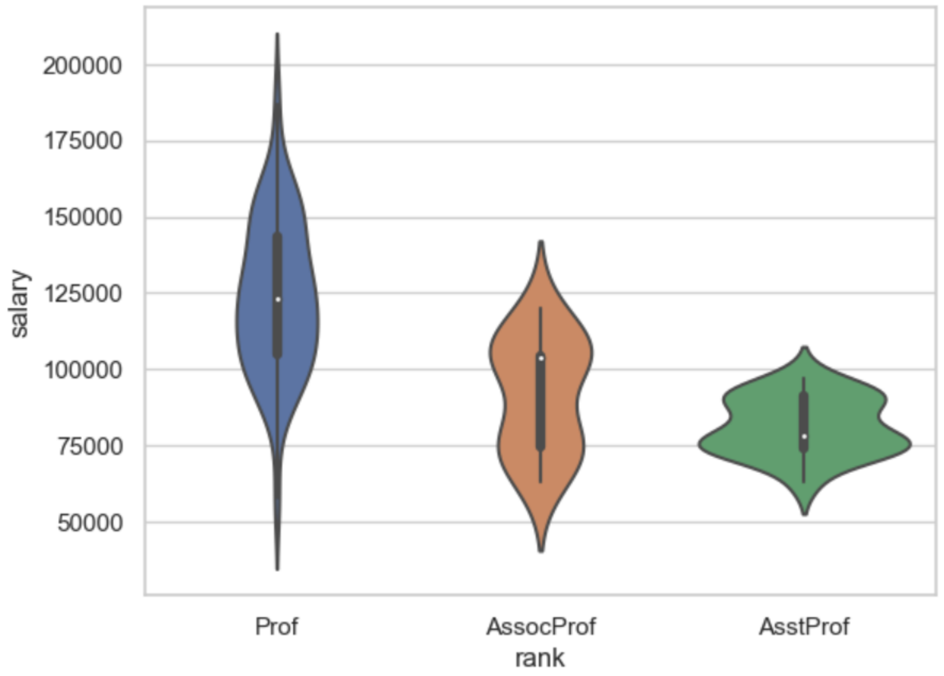
Remember: the formula provides the position in the ordered values, **but the value at that position is the result.** If position is between two entries, we take the average of those two values (as done at left).

Box and Whisker / Violin Plots

In [1]:
```
import pandas as pd
import seaborn as sns
df = pd.read_csv("./Salaries.csv")
sns.boxplot(x="rank",y="salary",data = df)
```
Out[1]: <AxesSubplot:xlabel='rank', ylabel='salary'>



In [2]:
```
sns.set(style = 'whitegrid')
sns.violinplot(x ="rank",
               y ="salary",
               data = df)
```
Out[2]: <AxesSubplot:xlabel='rank', ylabel='salary'>

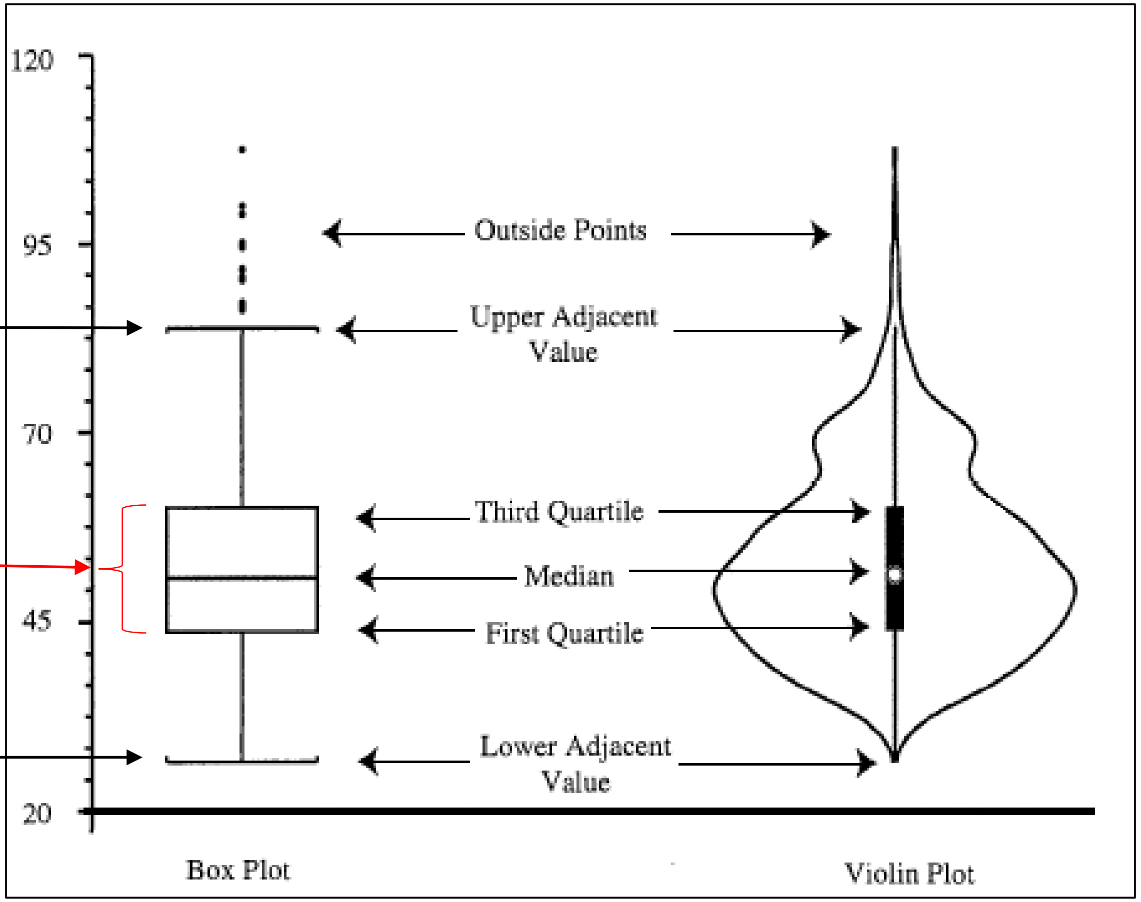


Box and Violin Plot Explanation

Highest value within 1.5 * IQR of third quartile (Q3)

Interquartile Range = third quartile minus first quartile (i.e., Q3-Q1)

Lowest value within 1.5 * IQR of first quartile (Q1)



The width of the violin plot at any given level is a representation of the frequency of observations with the value at that level. This is analogous to the height of a bar representing a bin in a histogram.

Note: the "whiskers" on the plots are not fixed at 1.5*IQR, they extend only to the farthest value found within that range.

Dispersion - Range

Unlike most of the other descriptive statistic measures we will discuss, pandas does not have a range() method built in. To some degree, this is a reflection on how often you can expect to use this particular characteristic.

However, it offers an opportunity to revisit some of the coding we discussed in the first session and practice applying a basic calculation in a loop with a conditional statement tied to some formatted output.

```
df.describe()
```

	yrs.since.phd	yrs.service	salary
count	78.000000	78.000000	78.000000
mean	19.705128	15.051282	108023.782051
std	12.498425	12.139768	28293.661022
min	1.000000	0.000000	57800.000000
25%	10.250000	5.250000	88612.500000
50%	18.500000	14.500000	104671.000000
75%	27.750000	20.750000	126774.750000
max	56.000000	51.000000	186960.000000

```
salary_range = df['salary'].max() - df['salary'].min()
print(salary_range)
```

```
129160
```

```
#using the kind property to find numeric columns to perform a range calculation on
for entry in df.columns:
    if(df[entry].dtype.kind in 'iufc'):
        curr_range = df[entry].max() - df[entry].min()
        print("Value range for {} is {}".format(entry,curr_range))
#Here, the kind property returns a character where:
#i denotes integer
#u denotes unsigned integer
#f denotes float
#c denotes complex numbers
```

```
Value range for yrs.since.phd is 55
Value range for yrs.service is 51
Value range for salary is 129160
```

Dispersion – Standard Deviation

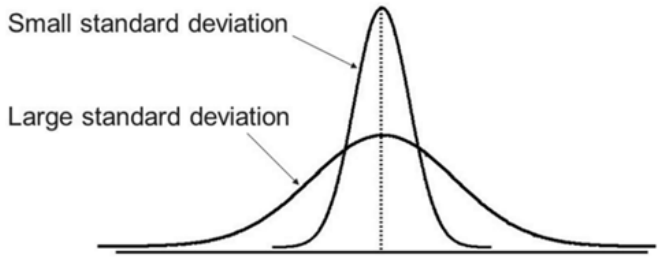
Variance is the ***average squared deviations from the mean***, while the <u>standard deviation is the square root of this number.</u>
Both measures reflect variability in a distribution, but their units differ.
Standard deviation is expressed in the same units as the original values.

Population

Variance = $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$

Mean = μ

Std. dev. = σ



In terms of the histogram …

Sample

Variance = $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Mean = $\bar{x}$

Std. dev. = s

Dispersion – Standard Deviation

Why do we use the standard deviation (SD), which is based on squared values?
Let’s use a small example data set to help explain.

	Sample Values			deviation from mean	absolute deviation	squared deviation
	5			-3.6	3.6	12.96
	8			-0.6	0.6	0.36
	9			0.4	0.4	0.16
	10			1.4	1.4	1.96
	11			2.4	2.4	5.76
sum	43		sum of deviations	0	8.4	21.2
mean	8.6		average deviation	0	1.68	4.24
						2.06

Variance

Standard Deviation (square root of variance)

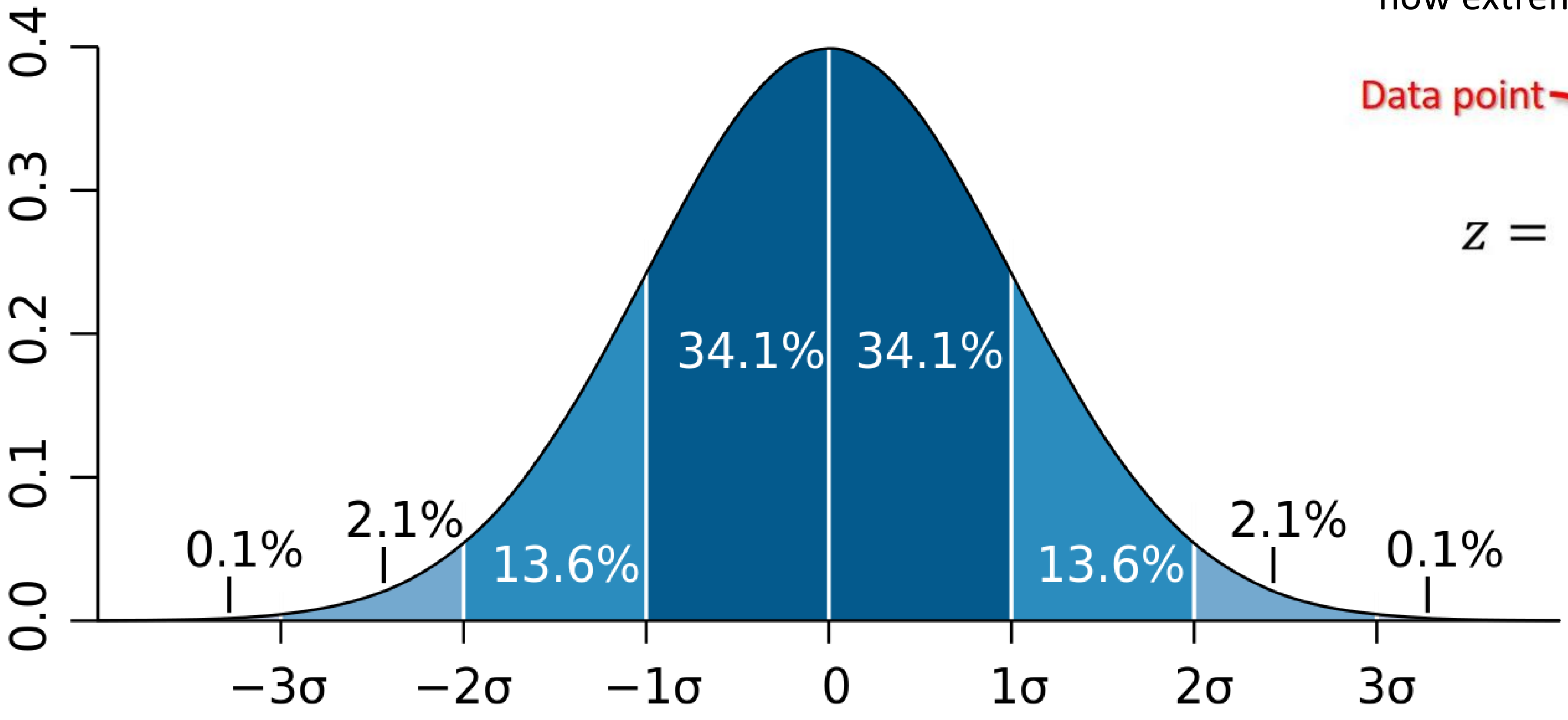
The average of actual deviations will be zero because their sum will always be zero.

Taking the absolute values would solve this aspect and there are good arguments that the average of absolute deviations(mean deviation - MD) is more intuitive and meaningful.

However, there are problems with differentiating an absolute value function and perhaps more importantly, it has been shown that the SD of a sample is a more consistent estimator of the SD of a population than the MD of a sample is of the MD of a population.

Standard Deviation Relationship With a Normal Distribution

Allows calculation of the Z-score, a very useful measure for how extreme a datapoint is.



Data point

Mean

$$z = \frac{(x - \mu)}{\sigma}$$

Standard deviation

Coefficient of Variation

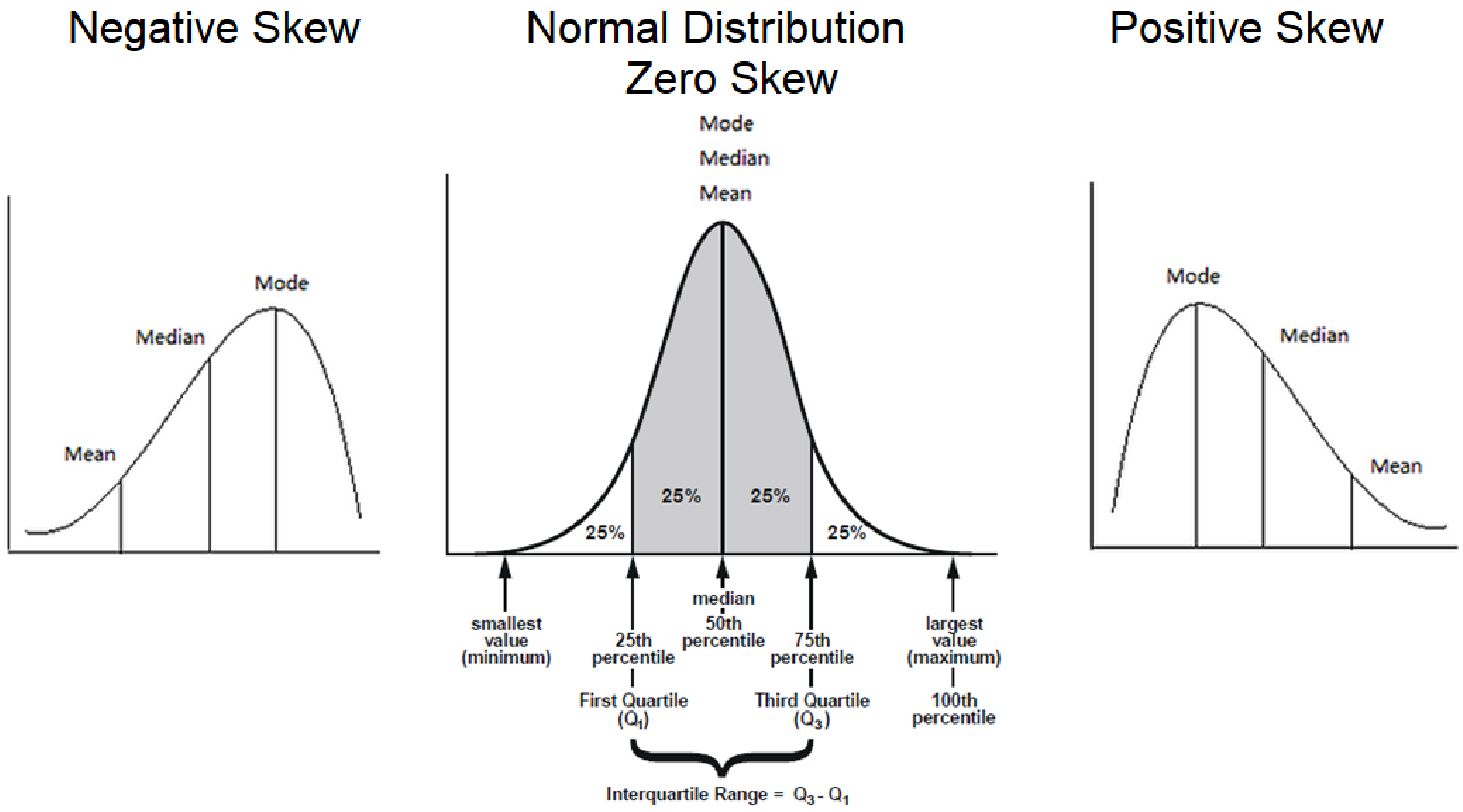
- Standard Deviation is a useful measure of variation **within** a dataset
- Comparing dispersion across different datasets using it may be problematic
  - Different units of measure
  - Even with same units, means may be quite different and have larger deviation values, but same proportional deviation
- A relative measure of variation is provided by the Coefficient of Variation:

$$C.V. = \frac{s}{\bar{x}}(100)$$

= standard deviation/mean * 100

The standard deviation and the mean are in the same units, which cancel out, providing a unitless proportional measure of variation

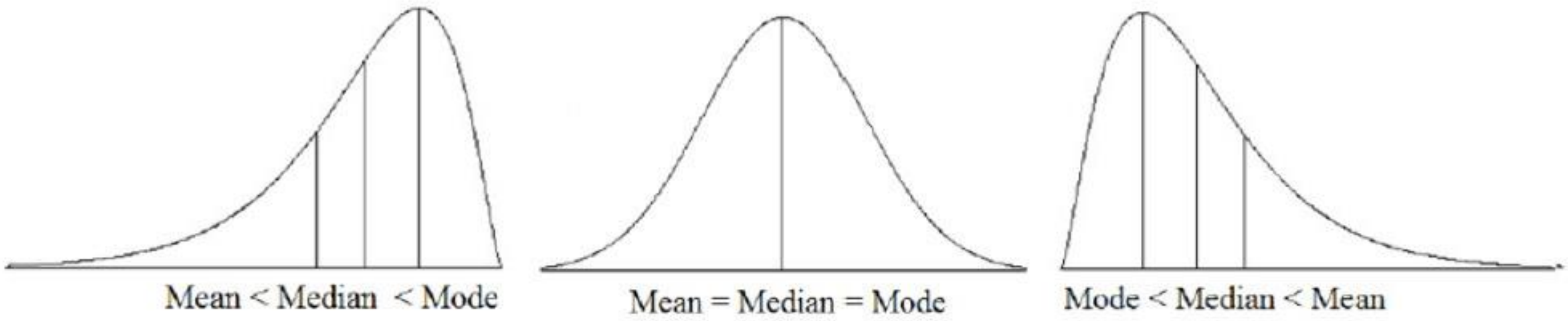
Dispersion – Skew
Effects on – Mean, Median and Mode



Calculating Skew

- There are multiple methods to calculate skew
- One commonly seen in texts is the "Alternative Pearson Mode Skewness":
  - Skew = 3 * (Mean – Median)/Standard Deviation
  - Pandas uses a substantially more complicated calculation that we are not going to review, but the concept is the same

Left Skew | Symmetric | Right Skew

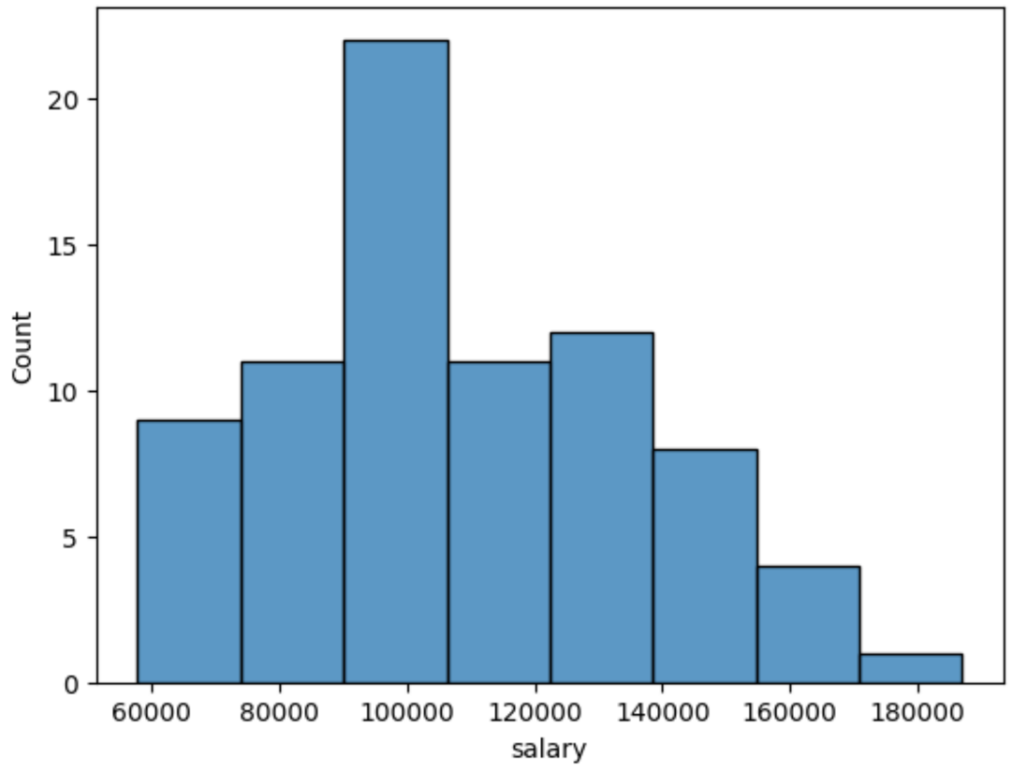


General Skew Values	Highly Skewed less than -1	Moderately Skewed -1 to -.5	Approximately Symmetric -.5 to .5	Moderately Skewed .5 to 1	Highly Skewed greater than 1

Calculating Skew in Pandas

```
import pandas as pd
import seaborn as sns
df = pd.read_csv("./Salaries.csv")
sns.histplot(df['salary'], kde=False)
```

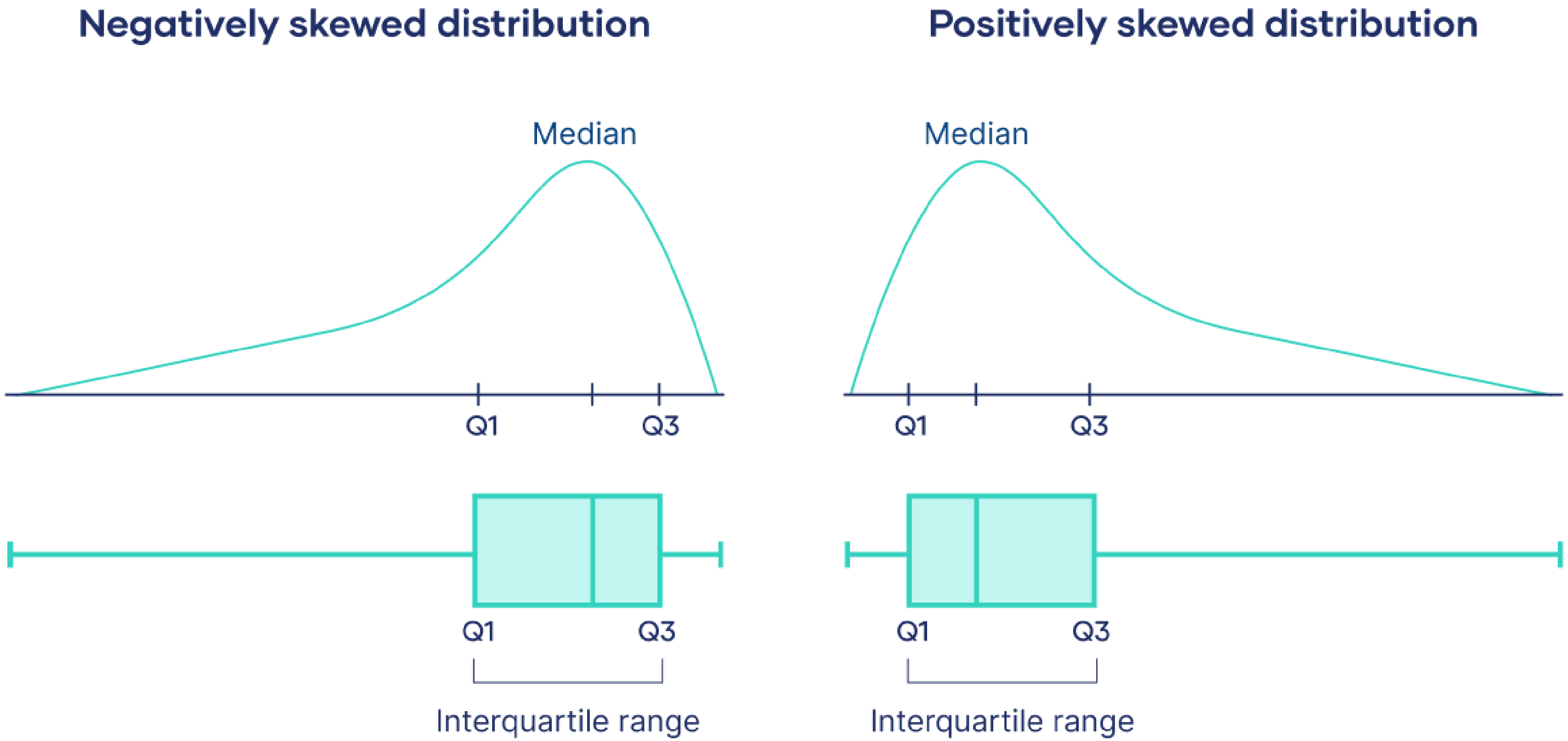
```
<AxesSubplot:xlabel='salary', ylabel='Count'>
```




```
df['salary'].skew()
```

```
0.45210256755588096
```

Visualizing Skew Via Box Plots



Dispersion - Kurtosis:
The propensity of data to have extreme values

Three types of distributions related to kurtosis:

- Platykurtic: flattest peak, highly dispersed
- Mesokurtic: medium peaked
- Leptokurtic: sharply peaked with fat tails, less variable

Don’t memorize this formula!

$$\text{Fisher's Kurtosis} = \sum_{i=1}^{N} \frac{\frac{X_i - \bar{X}}{N}}{S^4} - 3$$

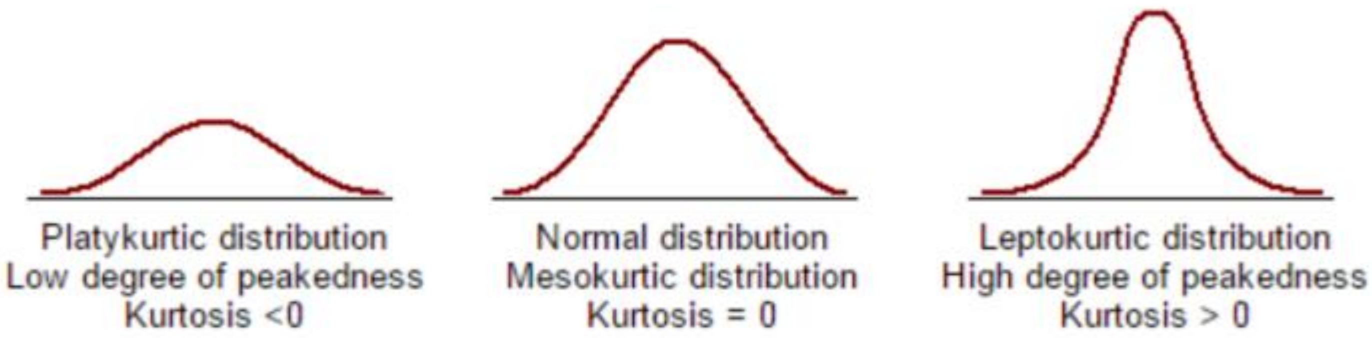
Where:

$\bar{X}$ is the mean

N is the sample size

S is the standard deviation

As with skew, there are multiple ways to calculate kurtosis. Panda’s uses Fisher’s Kurtosis (at left).



Correlation

- Correlation coefficient measures the strength of the linear relationship between two interval or ratio scale variables
- You might visualize such a relationship via a scatter plot, but the coefficient provides a quantitative weighting to the relationship
- As with other measures, there are multiple methods to calculate correlation. Pandas uses the Pearson correlation by default.

Don’t memorize formula!

Pearson Correlation Coefficient

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

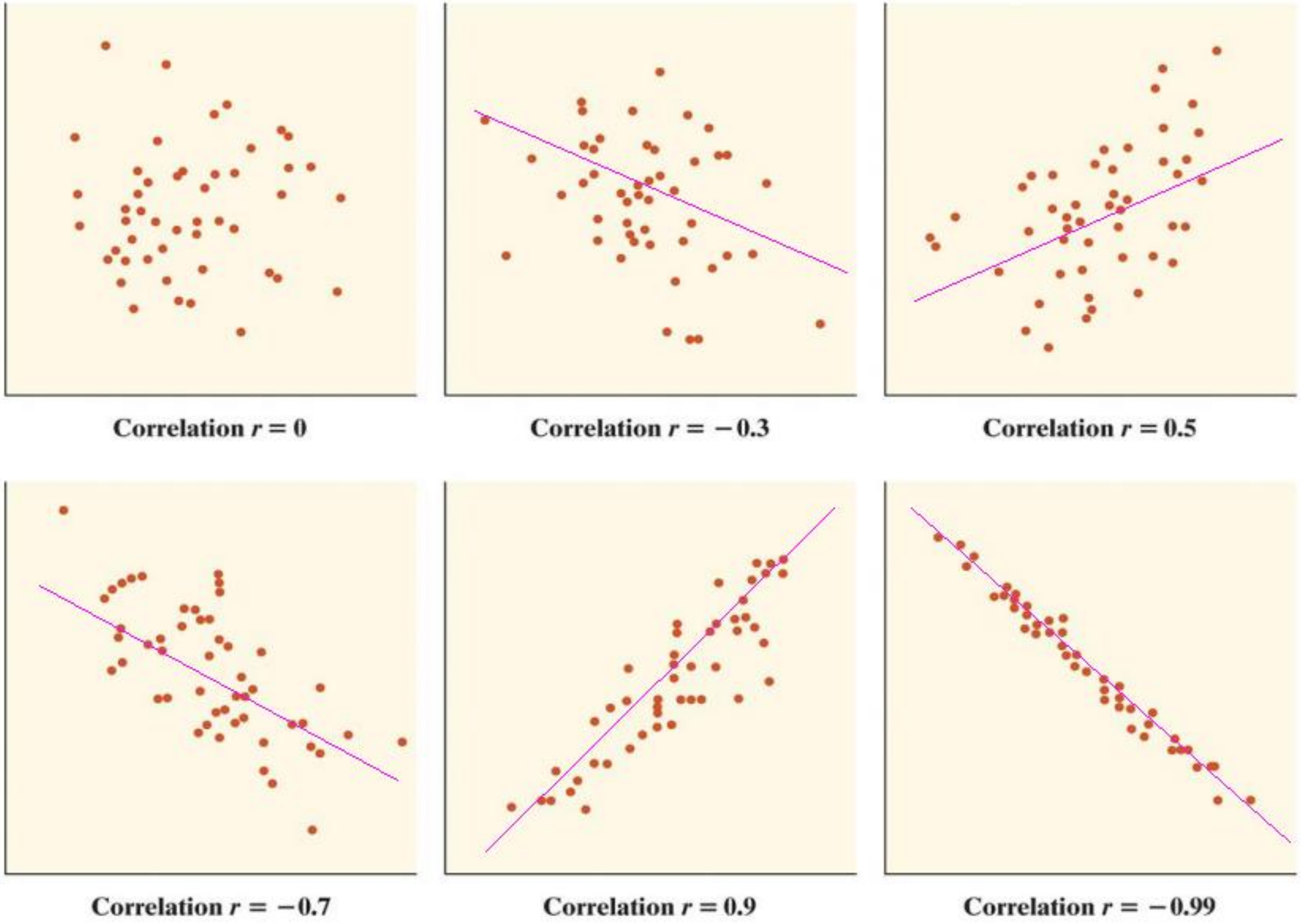
$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

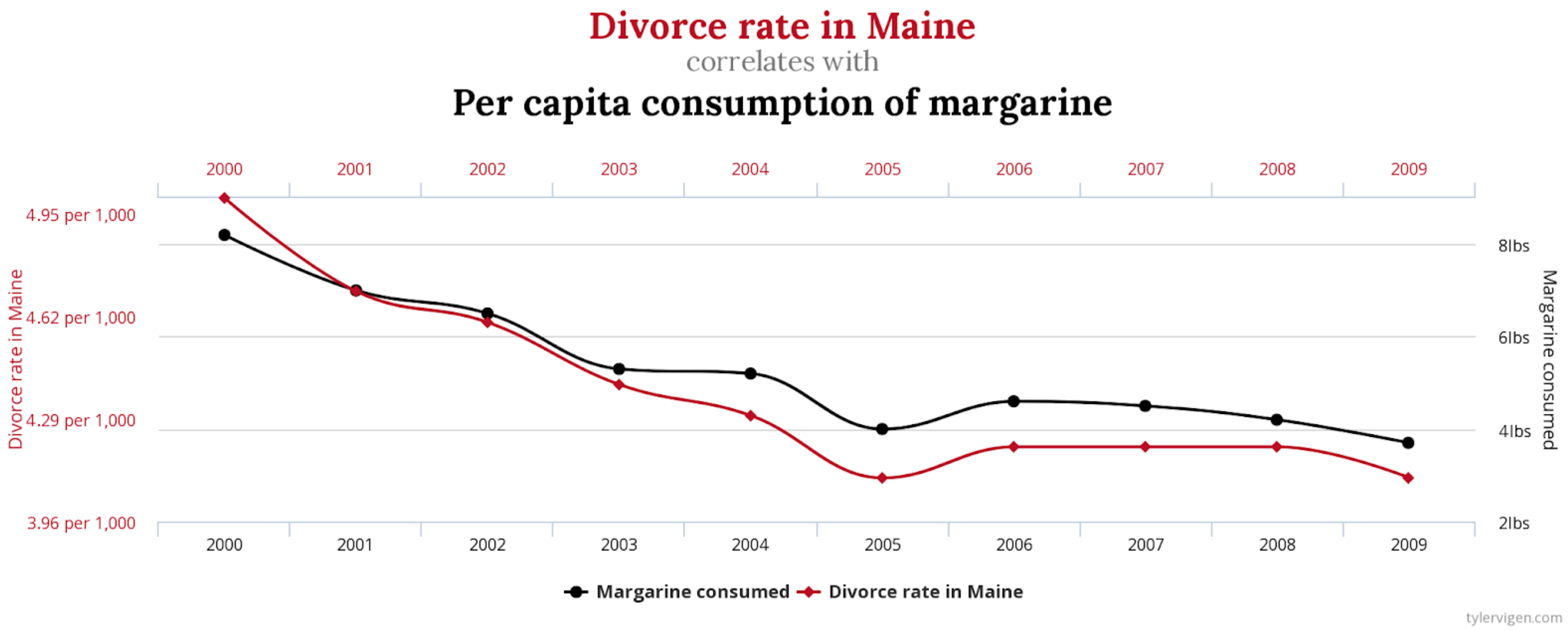
$\bar{y}$ = mean of the values of the y-variable

Visualizing Correlation

Note: the fit lines are a rough approximation

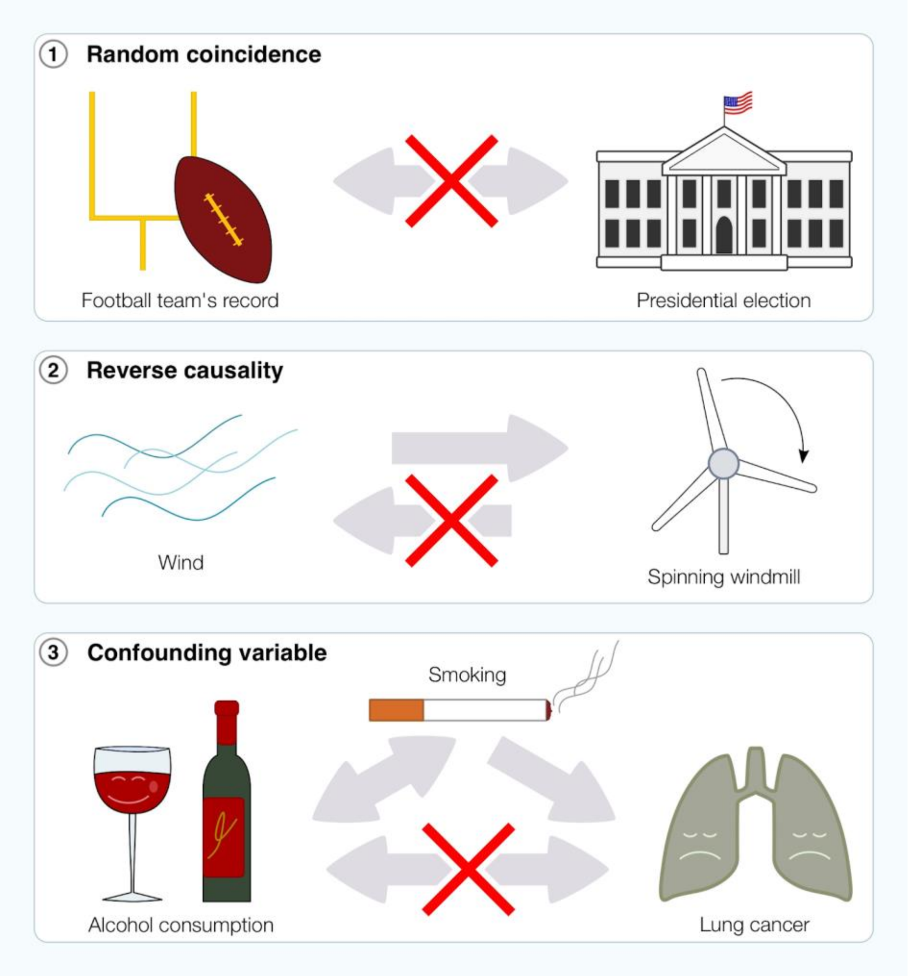


“Correlation Does Not Imply Causation”



From: https://sitn.hms.harvard.edu/flash/2021/when-correlation-does-not-imply-causation-why-your-gut-microbes-may-not-yet-be-a-silver-bullet-to-all-your-problems/

Common Themes of Inferring Causation From Correlation



From: https://sitn.hms.harvard.edu/flash/2021/when-correlation-does-not-imply-causation-why-your-gut-microbes-may-not-yet-be-a-silver-bullet-to-all-your-problems/

Viewing Correlation in Seaborn Heatmap

```
#plot a heat map of the correlation between numeric value sin the dataframe
sns.heatmap(df.corr(numeric_only=True),annot=True,cmap="RdYlGn")
```

```
<AxesSubplot:>
```

Note: sns refers to previously imported seaborn library and the “df” in this code is the dataframe variable loaded with the Salaries.csv data

Pima Indian (Native American) Diabetes Dataset

Column	Description	Range
Preg	Number of times pregnant	[0, 17]
Gluc	Plasma glucose concentration at 2 Hours in an oral glucose tolerance test (GTIT)	[0, 199]
BP	Diastolic Blood Pressure (mm Hg)	[0, 122]
Skin	Triceps skin fold thickness (mm)	[0, 99]
Insulin	2-Hour Serum insulin (µh/ml)	[0, 846]
BMI	Body mass index [weight in kg/(Height in m)]	[0, 67.1]
DPF	Diabetes pedigree function	[0.078, 2.42]
Age	Age (years)	[21, 81]
Outcome	Binary value indicating non-diabetic /diabetic	[0,1]

https://www.britannica.com/topic/Pima-people https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8943493/#CR5

Some Exploratory Data Analysis

- Lets take the concepts that we have covered use them to examine the Pima dataset
  - For one variable, we will examine
    - Frequency Distribution
    - Central Tendencies
    - Variance/Dispersion
    - Boxplot and Violin Plot
    - Correlation with other variables

First Step

- Remember, we need to import the libraries we will be using (pandas and seaborn) and load the data. Subsequent slides show code that assumes this has been done.

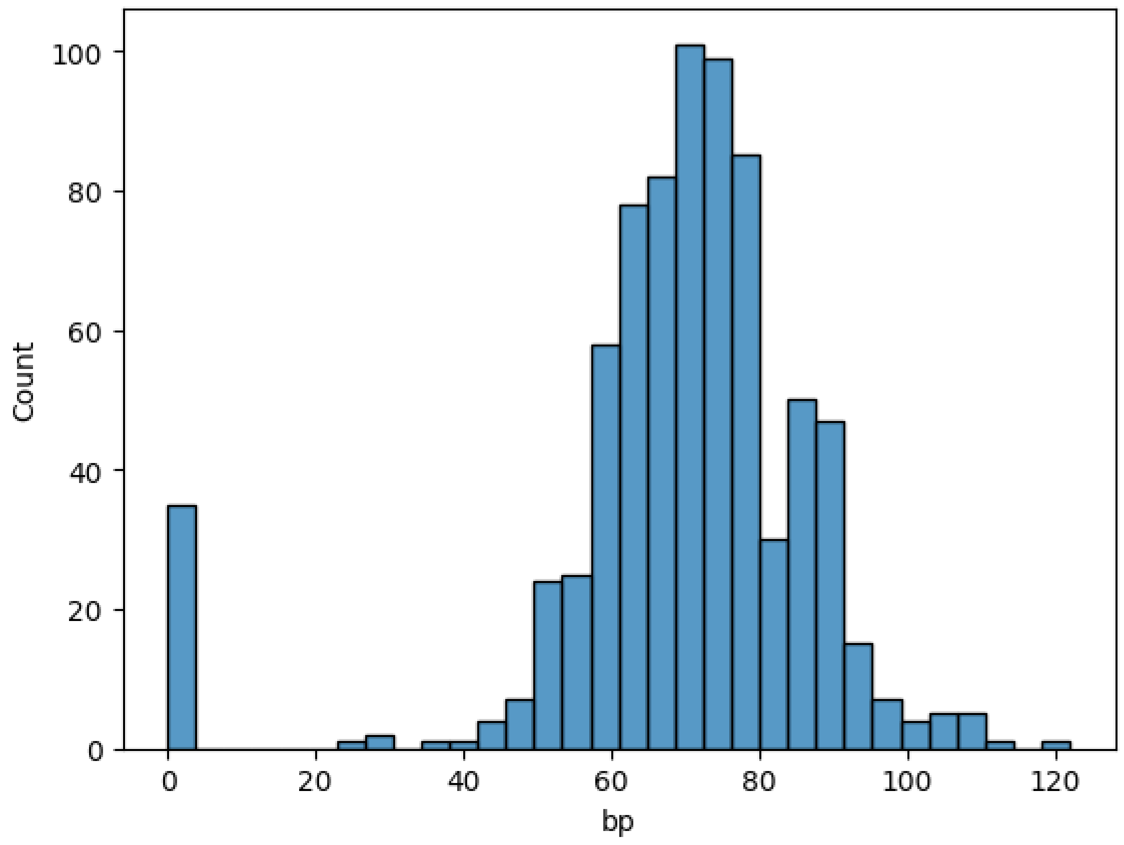
```
import pandas as pd
import seaborn as sns
#the following code uses the explicit path to the file ON MY SYSTEM!!! You need to know where you put the file on YOUR system
#and use the appropriate information (including path format for your system type...Windows/Mac/Linux)
#a linux example: filename  = "/home/fd299212/dat/pima-diabetes.data.csv"
#example using a relative file path: filename = "./pima-diabetes.data.csv"

filename = "C:\\Users\\doylef\\Desktop\\NNSE_784\\course_lectures\\data\\health\\pima-diabetes.data.csv"
df = pd.read_csv(filename)
```

Blood Pressure Histogram

```
sns.histplot(df['bp'], kde=False)
```

```
<AxesSubplot:xlabel='bp', ylabel='Count'>
```



Blood Pressure describe() Output

```
df['bp'].describe()
```

```
count    768.000000
mean      69.105469
std       19.355807
min        0.000000
25%       62.000000
50%       72.000000
75%       80.000000
max      122.000000
Name: bp, dtype: float64
```

```
df['bp'].mode()
```

```
0    70
dtype: int64
```

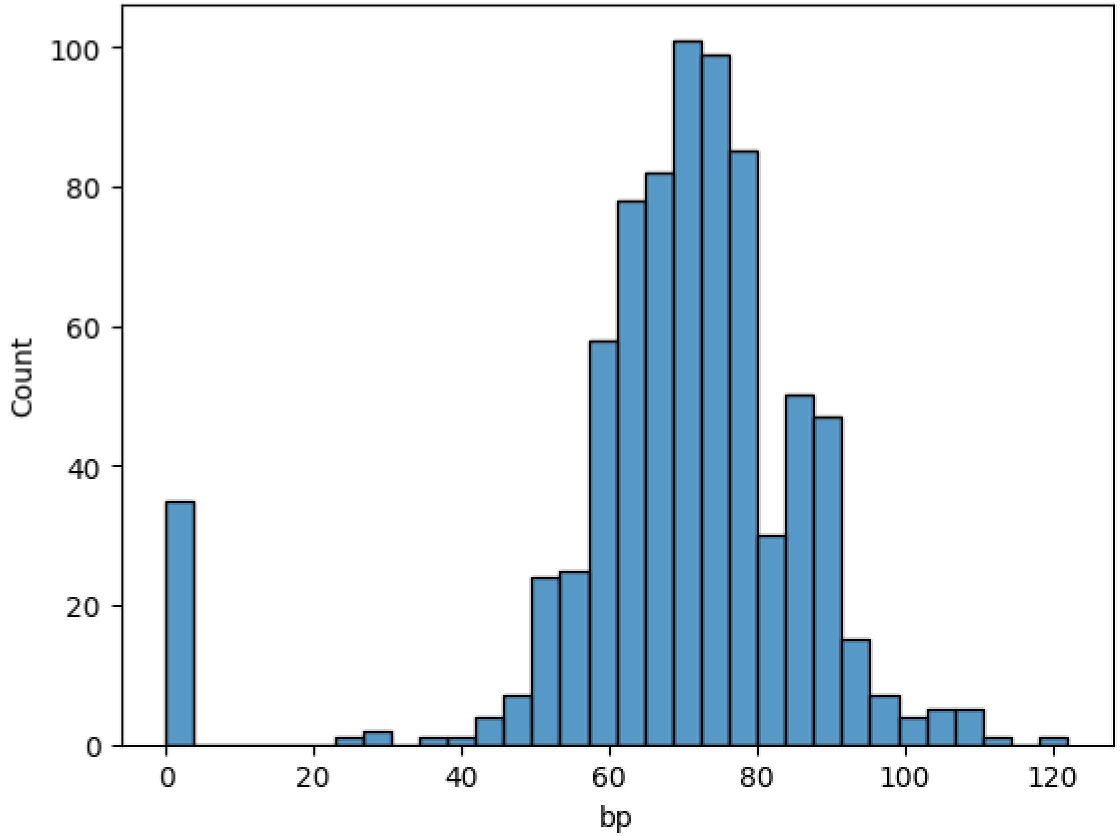
Blood Pressure
Skew and Kurtosis

```
df['bp'].skew()
```

-1.8436079833551302

```
df['bp'].kurtosis()
```

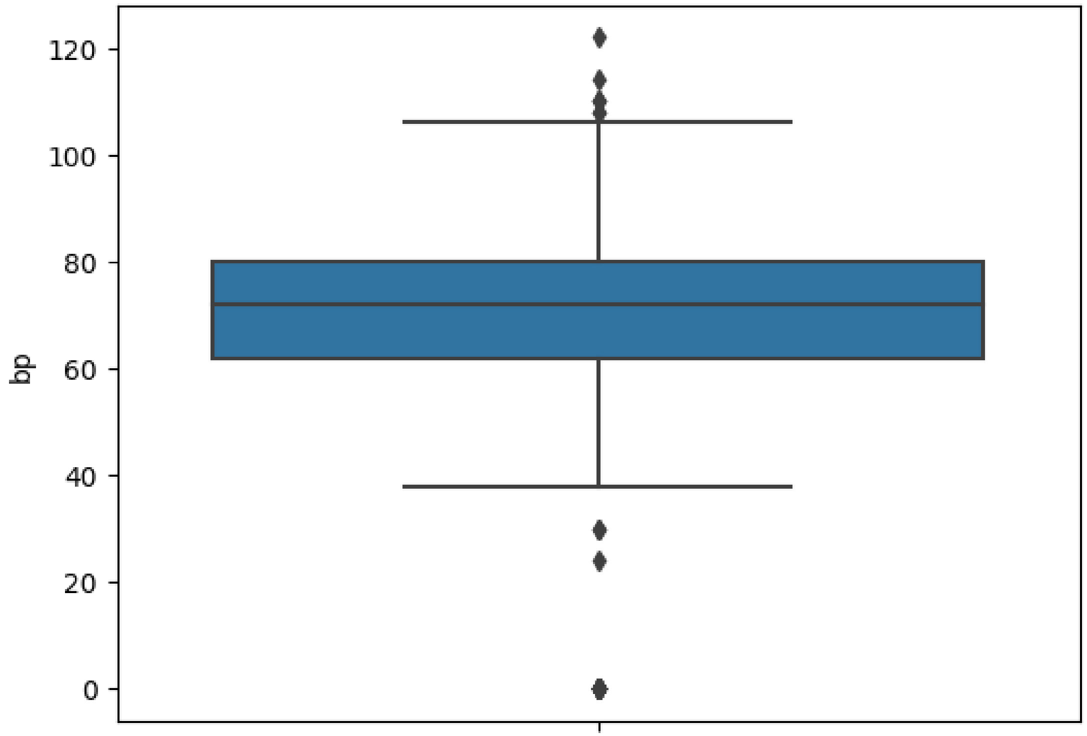
5.180156560082496



Blood Pressure – Box Plot

```
sns.boxplot(y="bp",data = df)
```

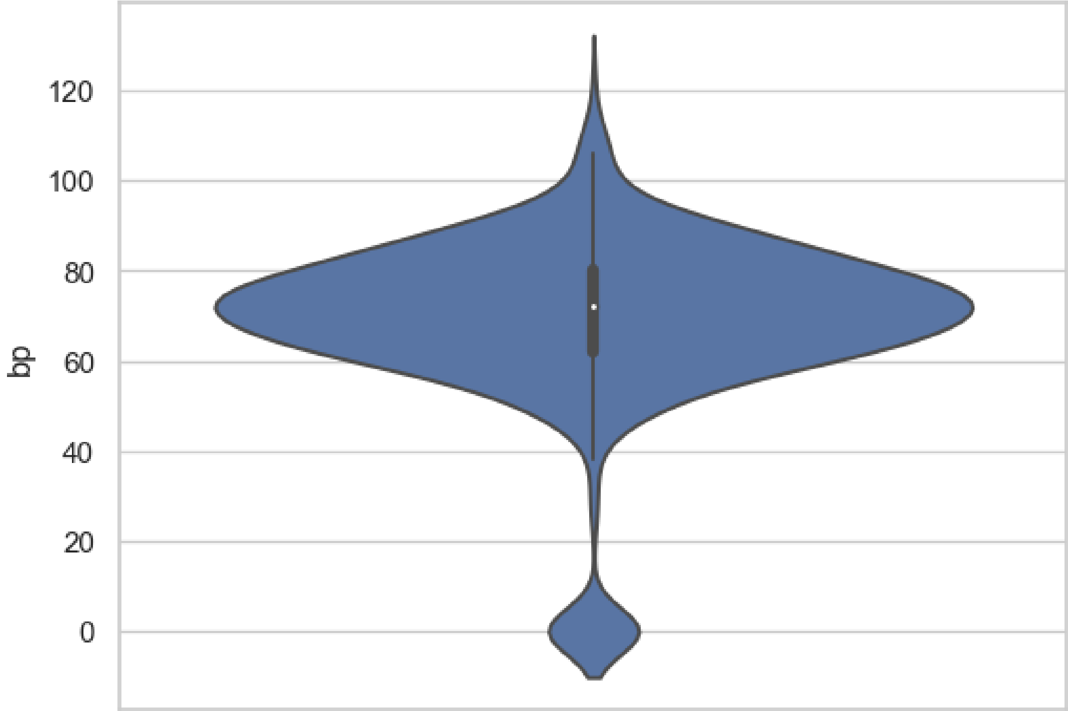
```
<AxesSubplot:ylabel='bp'>
```



Blood Pressure – Violin Plot

```
sns.set(style = 'whitegrid')
sns.violinplot( y ="bp",data = df)
```

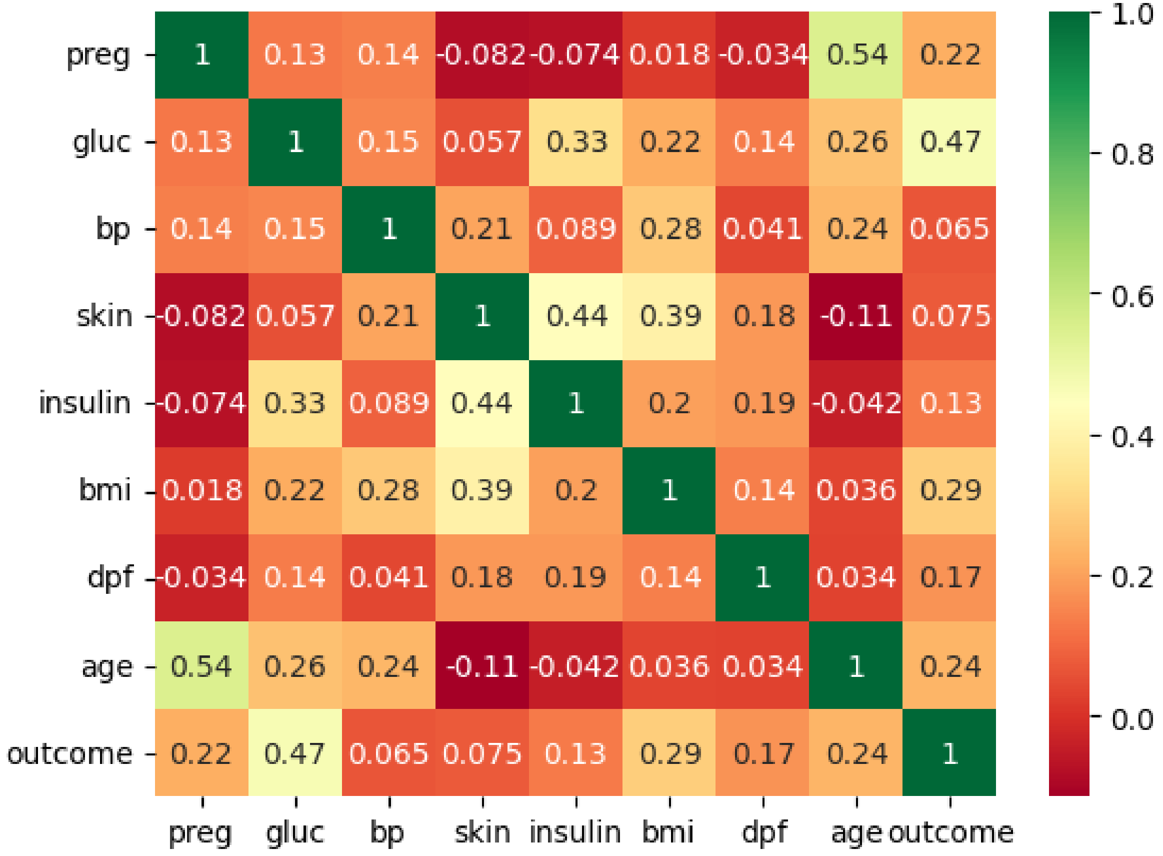
<AxesSubplot:ylabel='bp'>



Heatmap for Pima Diabetes Dataset

```
sns.heatmap(df.corr(numeric_only=True),annot=True,cmap="RdYlGn")
```

```
<AxesSubplot:>
```



Note: depending on the version of pandas you are using, the “numeric_only=True may cause problems. If you see an error regarding it, remove it from the parantheses (but leave the empty parantheses as this is a method call)