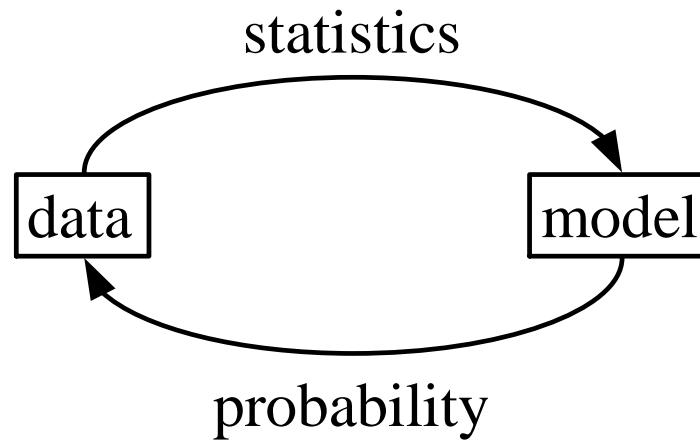




Statistics for Machine Learning

Prof. Seungchul Lee
Industrial AI Lab.

Statistics and Probability



Populations and Samples

- A **population** includes all the elements from a set of data
- A **parameter** is a quantity computed from a population
 - mean, μ
 - variance, σ^2
- A **sample** is a subset of the population.
 - one or more observations
- A **statistic** is a quantity computed from a sample
 - sample mean, \bar{x}
 - sample variance, s^2
 - sample correlation, S_{xy}

How to Generate Random Numbers

- Data sampled from population/process/generative model

```
## random number generation (1D)
m = 1000;

# uniform distribution  $U(0,1)$ 
x1 = np.random.rand(m,1);

# uniform distribution  $U(a,b)$ 
a = 1;
b = 5;
x2 = a + (b-a)*np.random.rand(m,1);

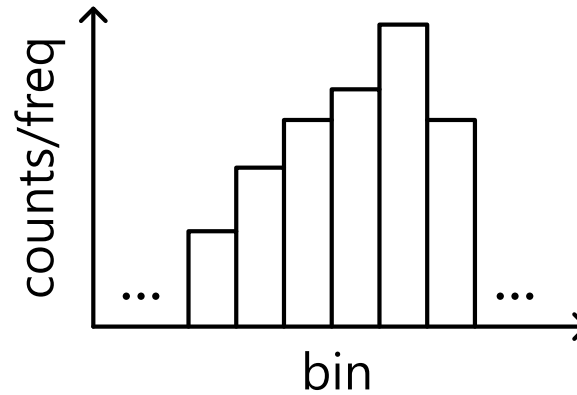
# standard normal (Gaussian) distribution  $N(0,1^2)$ 
# x3 = np.random.normal(0, 1, m)
x3 = np.random.randn(m,1);

# normal distribution  $N(5,2^2)$ 
x4 = 5 + 2*np.random.randn(m,1);

# random integers
x5 = np.random.randint(1, 6, size = (1,m));
```

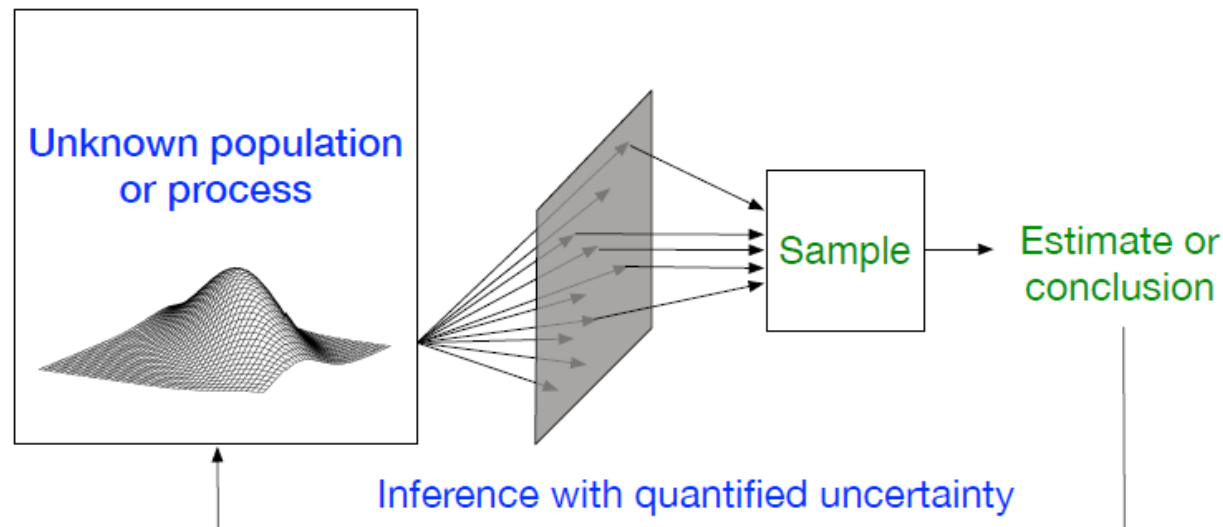
Histogram

- Graphical representation of data distribution
⇒ rough sense of density of data

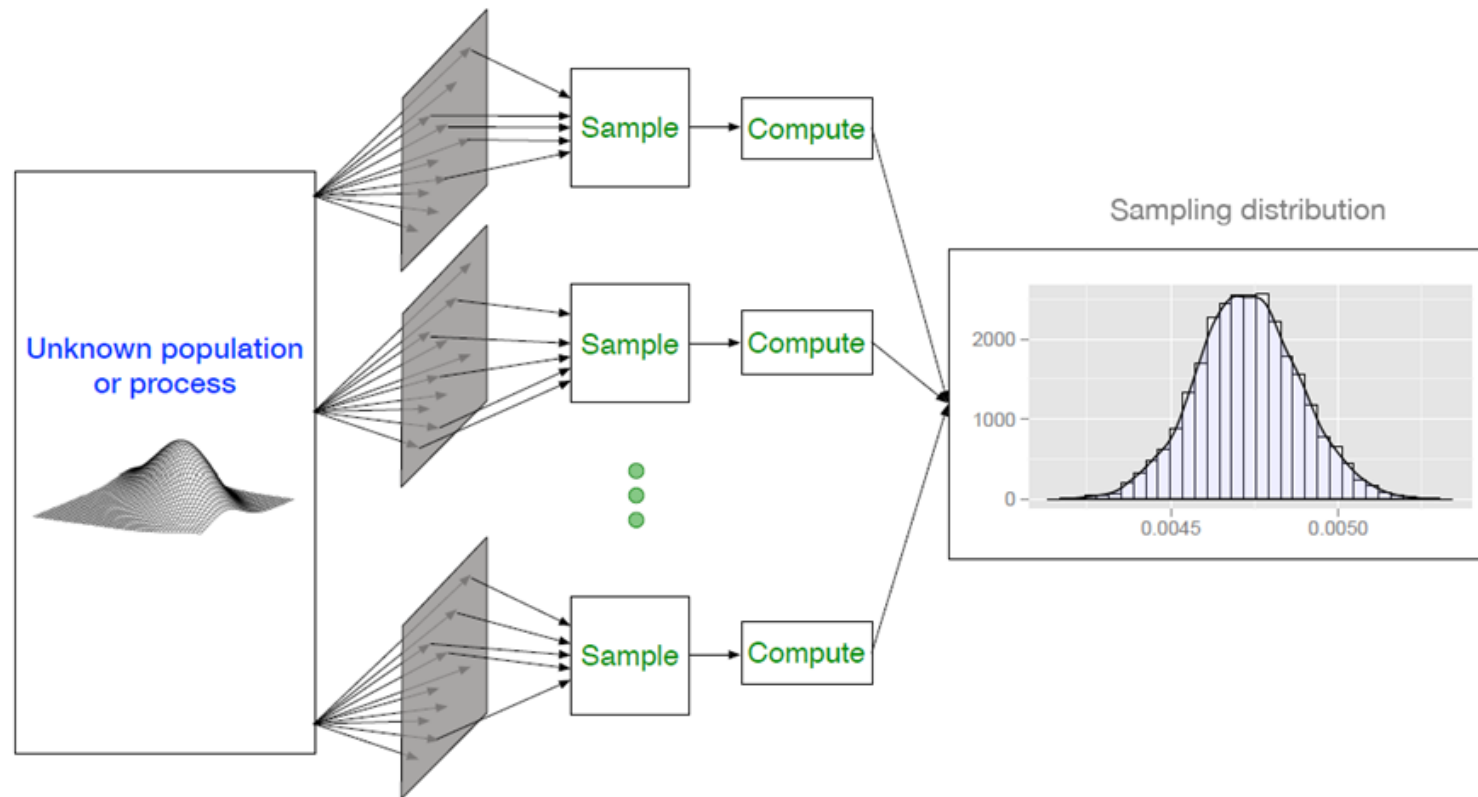


Inference

- True population or process is modeled probabilistically
- Sampling supplies us with realizations from probability model
- Compute something, but recognize that we could have just as easily gotten a different set of realizations

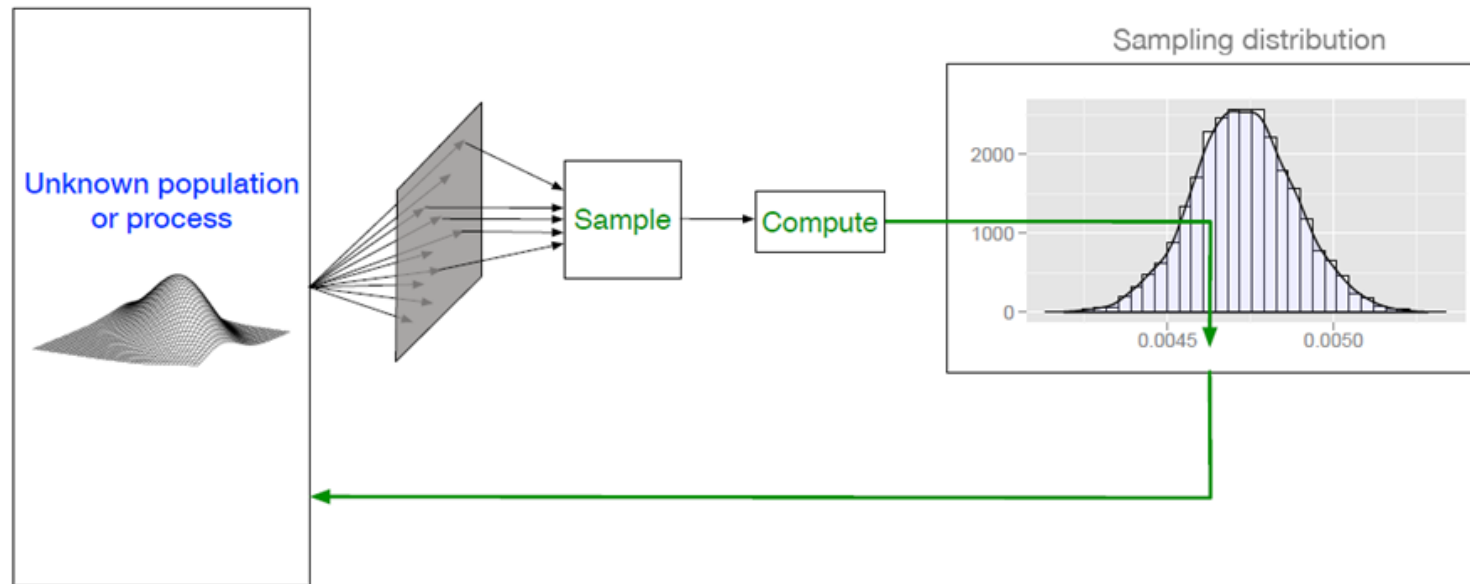


Inference



Inference

- We want to infer the characteristics of the true probability model from our one sample.

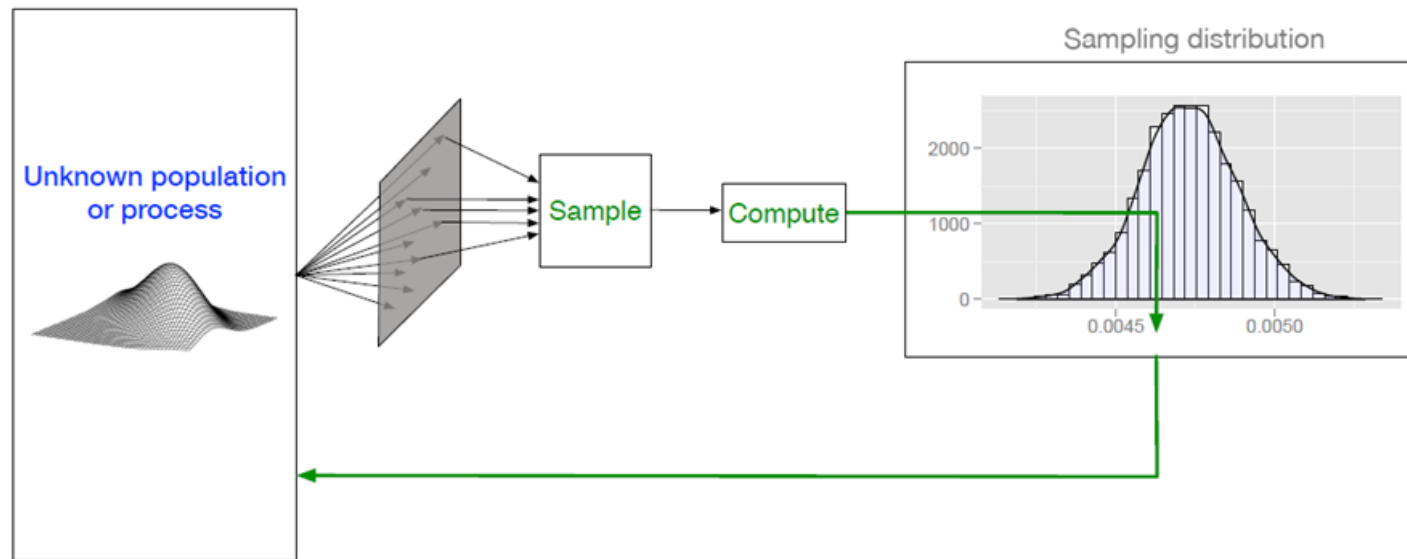


The Law of Large Numbers

- Sample mean converges to the population mean as sample size gets large

$$\bar{x} \rightarrow \mu_x \quad \text{as} \quad m \rightarrow \infty$$

- True for any probability density functions



Sample Mean and Sample Size

- Sample mean and sample variance

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_m}{m}$$
$$s^2 = \frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m - 1}$$

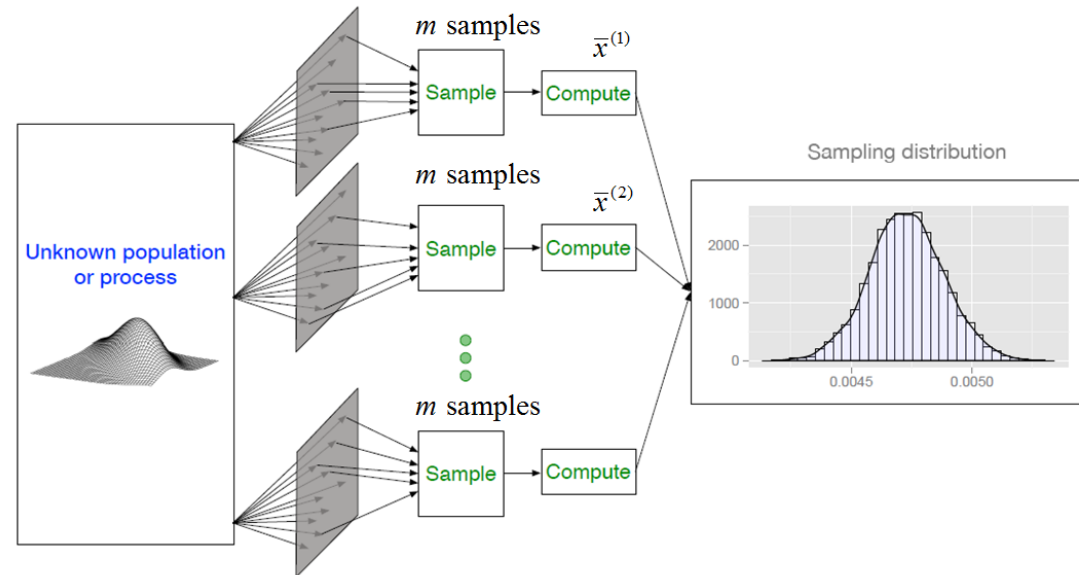
The Central Limit Theorem

- **Sample mean** (not samples) will be approximately normally distributed as a sample size $m \rightarrow \infty$

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_m}{m}$$

- More samples provide more confidence (or less uncertainty)
- Note: true regardless of any distributions of population

$$\bar{x} \rightarrow N\left(\mu_x, \left(\frac{\sigma}{\sqrt{m}}\right)^2\right)$$



Uniform Distribution: $x \sim U[0, 1]$

```
# statistics
# numerically understand statistics

m = 100
x = np.random.rand(m,1)

#xbar = 1/m*np.sum(x, axis = 0)
#np.mean(x, axis = 0)
xbar = 1/m*np.sum(x)
np.mean(x)

varbar = (1/(m - 1))*np.sum((x - xbar)**2)
np.var(x)

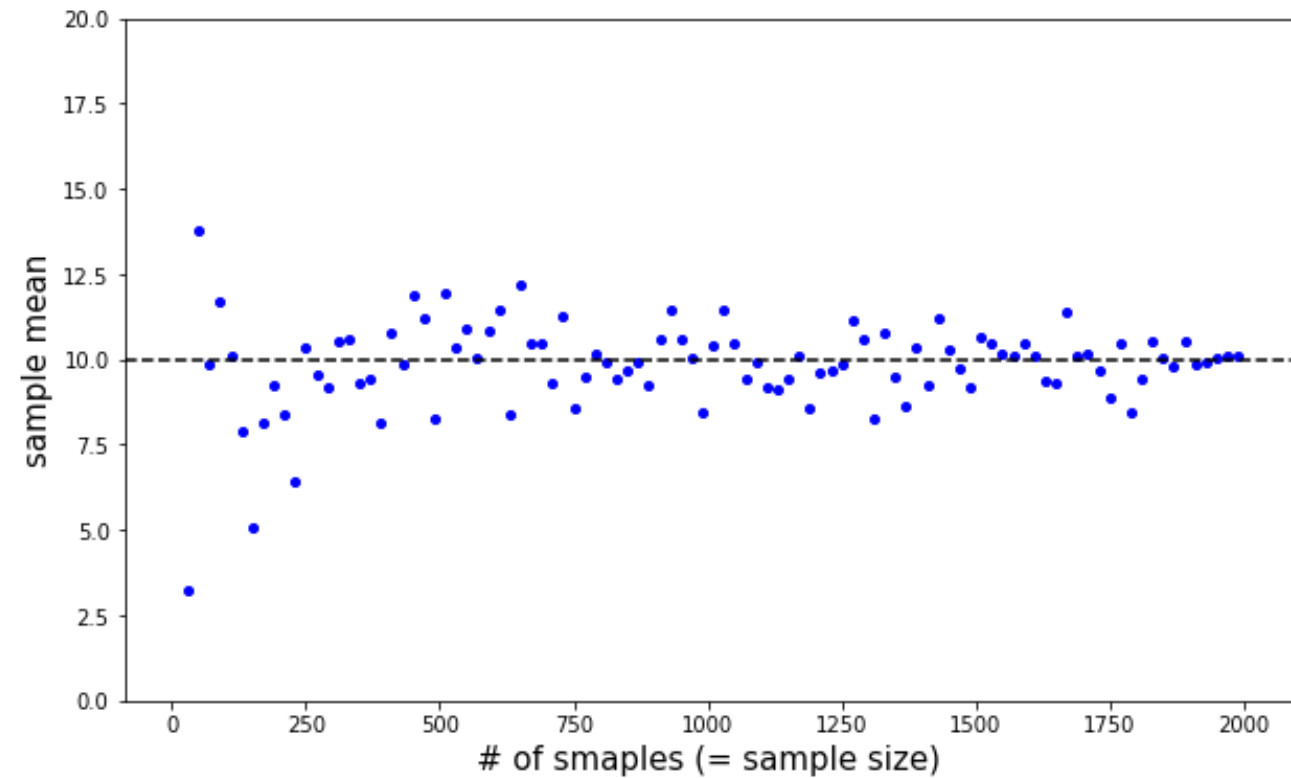
print(xbar)
print(np.mean(x))
print(varbar)
print(np.var(x))
```

```
0.5082513375791726
0.5082513375791726
0.10190494538417319
0.10088589593033145
```

Sample Size

```
# various sample size m
m = np.arange(10, 2000, 20)
means = []

for i in m:
    x = np.random.normal(10, 30, i)
    means.append(np.mean(x))
```



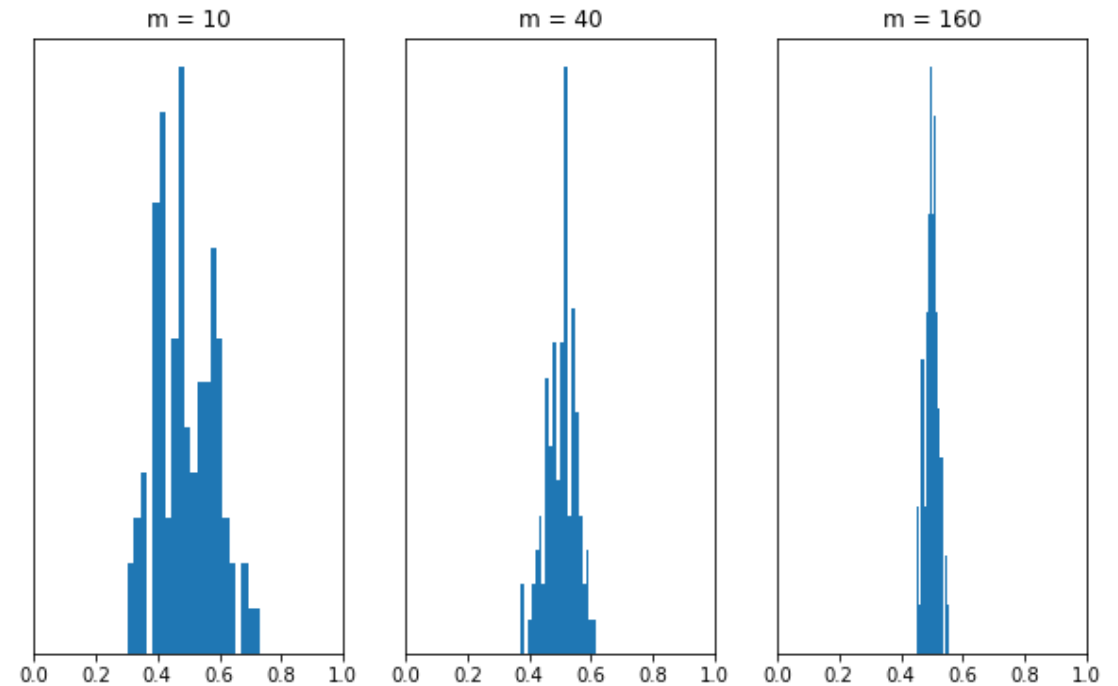
Variance Gets Smaller as m is Larger

- Seems approximately Gaussian distributed
- Numerically demonstrate that sample mean follows Gaussian distribution

```
N = 100
m = np.array([10, 40, 160]) # sample of size m

S1 = [] # sample mean (or sample average)
S2 = []
S3 = []

for i in range(N):
    S1.append(np.mean(np.random.rand(m[0], 1)))
    S2.append(np.mean(np.random.rand(m[1], 1)))
    S3.append(np.mean(np.random.rand(m[2], 1)))
```



Multivariate Statistics

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \end{bmatrix}, \quad X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & \\ - & (x^{(m)})^T & - \end{bmatrix}$$

- m observations $(x^{(1)}, x^{(2)}, \dots, x^{(m)})$

$$\text{sample mean } \bar{x} = \frac{x^{(1)} + x^{(2)} + \dots + x^{(m)}}{m} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\text{sample variance } S^2 = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \bar{x})^2$$

$$(\text{Note: population variance } \sigma^2 = \frac{1}{N} \sum_{i=1}^N (x^{(i)} - \mu)^2)$$

Correlation of Two Random Variables

$$\text{Sample Variance : } S_x = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \bar{x})^2$$

$$\text{Sample Covariance : } S_{xy} = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \bar{x}) (y^{(i)} - \bar{y})$$

$$\text{Sample Covariance matrix : } S = \begin{bmatrix} S_x & S_{xy} \\ S_{yx} & S_y \end{bmatrix}$$

$$\text{sample correlation coefficient : } r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$$

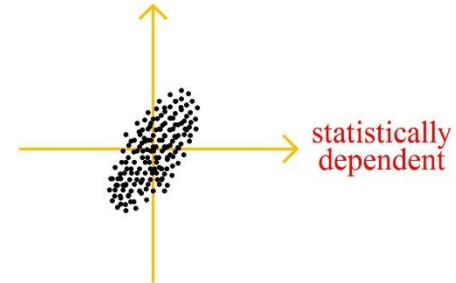
- Correlation
 - Strength of **linear** relationship between two variables, x and y

Correlation of Two Random Variables

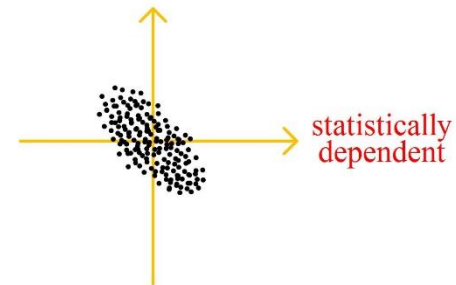
- Assume

$$x_1 \leq x_2 \leq \dots \leq x_n$$
$$y_1 \leq y_2 \leq \dots \leq y_n$$

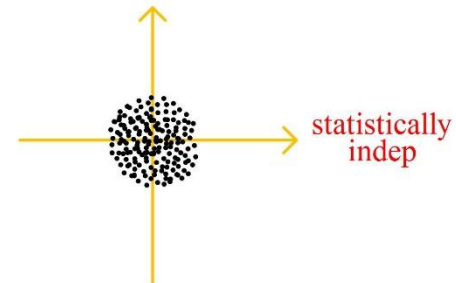
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}, \dots, \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$



$$\begin{bmatrix} x_1 \\ y_n \end{bmatrix}, \begin{bmatrix} x_2 \\ y_{n-1} \end{bmatrix}, \dots, \begin{bmatrix} x_n \\ y_1 \end{bmatrix}$$



$$\begin{bmatrix} x_i \\ y_j \end{bmatrix} \text{ random selection}$$



Correlation Coefficient

- $+1 \rightarrow$ close to a straight line
- $-1 \rightarrow$ close to a straight line
- Indicate how close to a **linear** line, but
- No information on slope

$$\begin{array}{ccc} 0 & \leq | \text{correlation coefficient} | & \leq 1 \\ \leftarrow & & \rightarrow \\ \text{(uncorrelated)} & & \text{(linearly correlated)} \end{array}$$

- Does not tell anything about causality

Correlation Coefficient

```
# correlation coefficient
```

```
m = 300
```

```
x = np.random.rand(m)
```

```
y = np.random.rand(m)
```

```
xo = np.sort(x)
```

```
yo = np.sort(y)
```

```
yor = -np.sort(-y)
```

```
plt.figure(figsize = (8, 8))
```

```
plt.plot(x, y, 'ko', label = 'random')
```

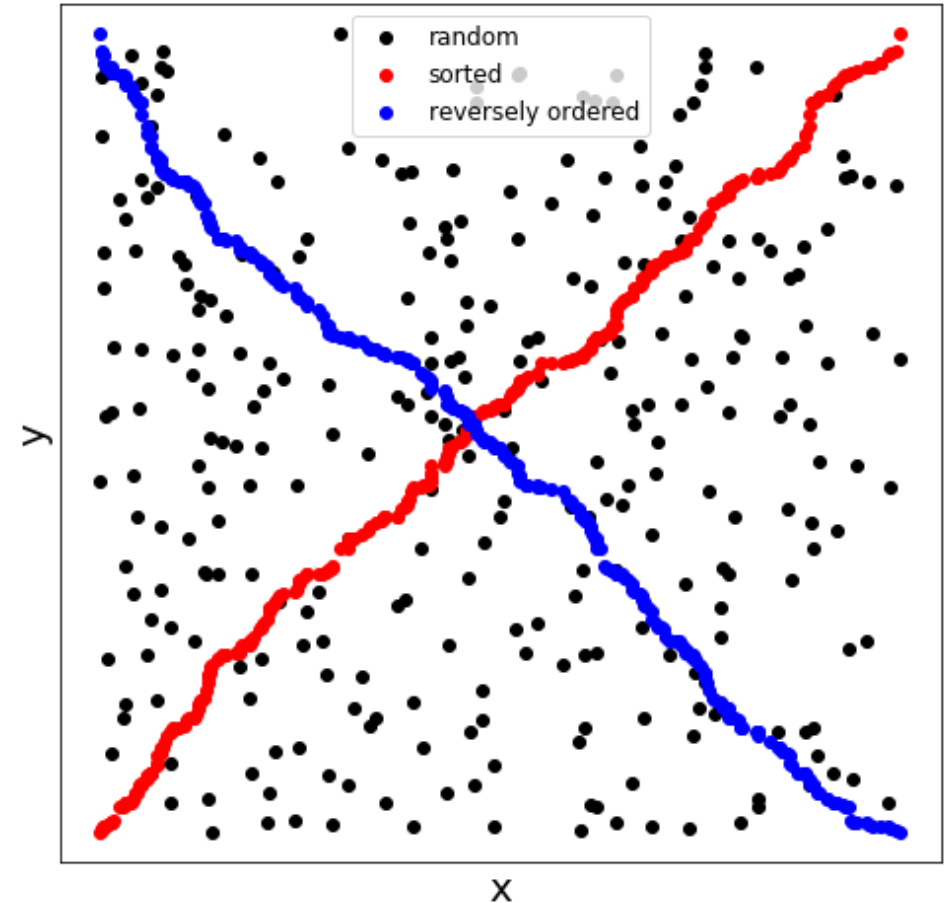
```
plt.plot(xo, yo, 'ro', label = 'sorted')
```

```
plt.plot(xo, yor, 'bo', label = 'reversely ordered')
```

```
[[ 1.          -0.01391645]  
 [-0.01391645  1.          ]]
```

```
[[1.          0.99745732]  
 [0.99745732  1.          ]]
```

```
[[ 1.          -0.99592996]  
 [-0.99592996  1.          ]]
```



Correlation Coefficient

```
# correlation coefficient

m = 300
x = 2*np.random.randn(m)
y = np.random.randn(m)

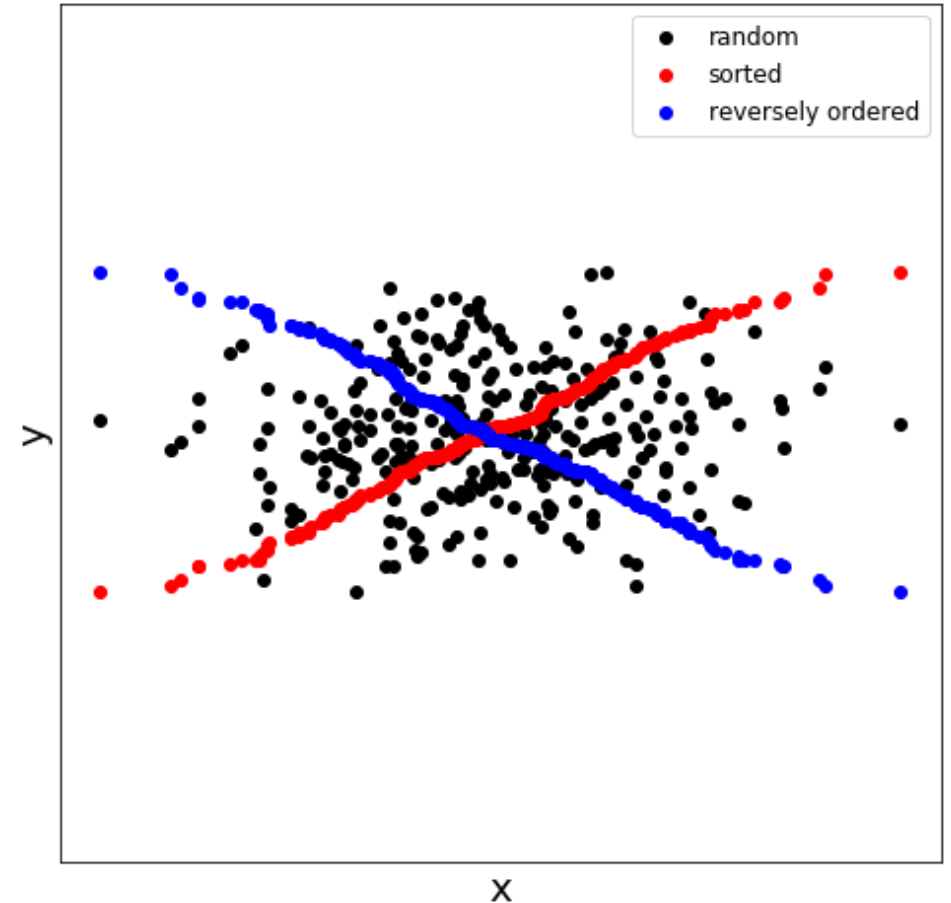
xo = np.sort(x)
yo = np.sort(y)
yor = -np.sort(-y)

plt.figure(figsize = (8, 8))
plt.plot(x, y, 'ko', label = 'random')
plt.plot(xo, yo, 'ro', label = 'sorted')
plt.plot(xo, yor, 'bo', label = 'reversely ordered')
```

```
[[1.          0.09583864]
 [0.09583864 1.          ]]
```

```
[[1.          0.9963007]
 [0.9963007 1.          ]]
```

```
[[ 1.          -0.99518884]
 [-0.99518884 1.          ]]
```



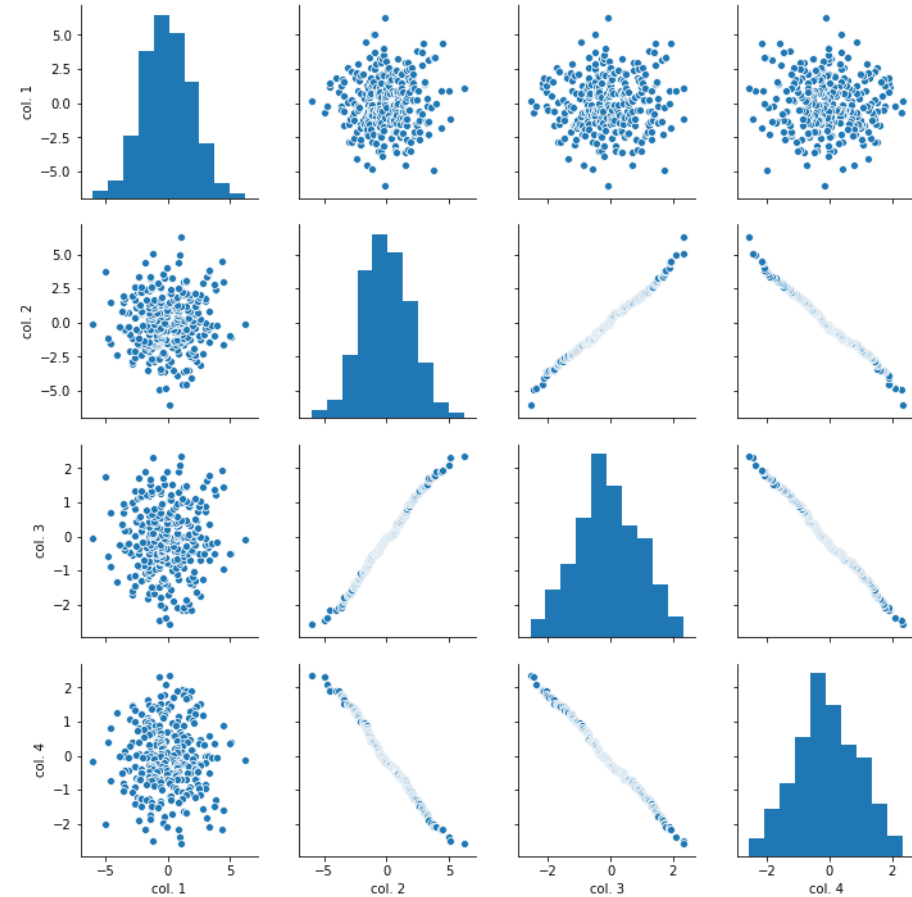
Correlation Coefficient Plot

- Plots correlation coefficients among pairs of variables
- <http://rpsychologist.com/d3/correlation/>

```
import seaborn as sns
import pandas as pd

d = {'col. 1': x, 'col. 2': xo, 'col. 3': yo, 'col. 4': yor}
df = pd.DataFrame(data = d)

sns.pairplot(df)
plt.show()
```



Covariance Matrix

$$\Sigma = \begin{bmatrix} E[(X_1 - \mu_1)(X_1 - \mu_1)] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & E[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & E[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}$$