Statistical Distribution

MET CS Generative AI

Reza Rawassizadeh

Statistics Definitions

Descriptive: It is used to describe the characteristics of the data, identifying narratives in the dataset. In particular, it enables us to draw some conclusions about the dataset.

Inferential: It is used to make inferences from data and find something from the data, which is the same reason we are using machine learning.

Statistics Definitions

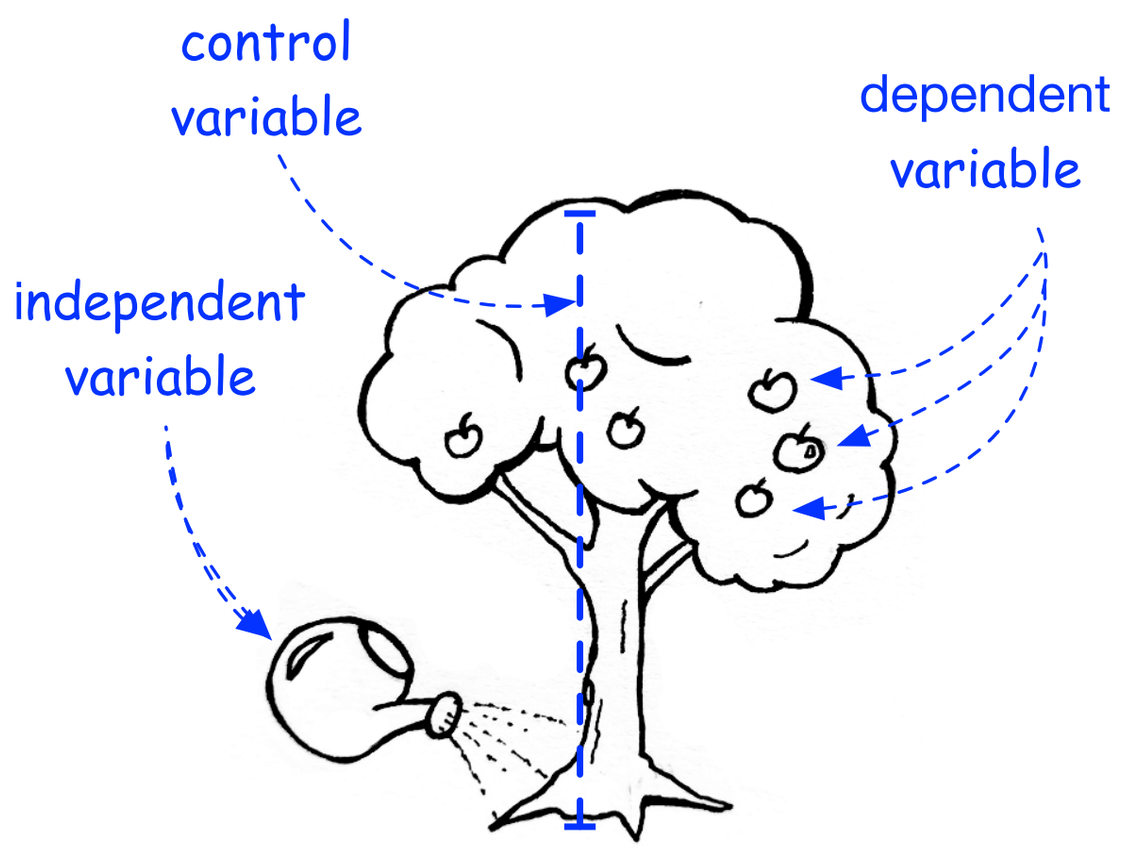
- **Variable** vs **Value**

- Value: **Continuous** vs **Discrete**

- Variables: **Categorical (nominal), numerical**

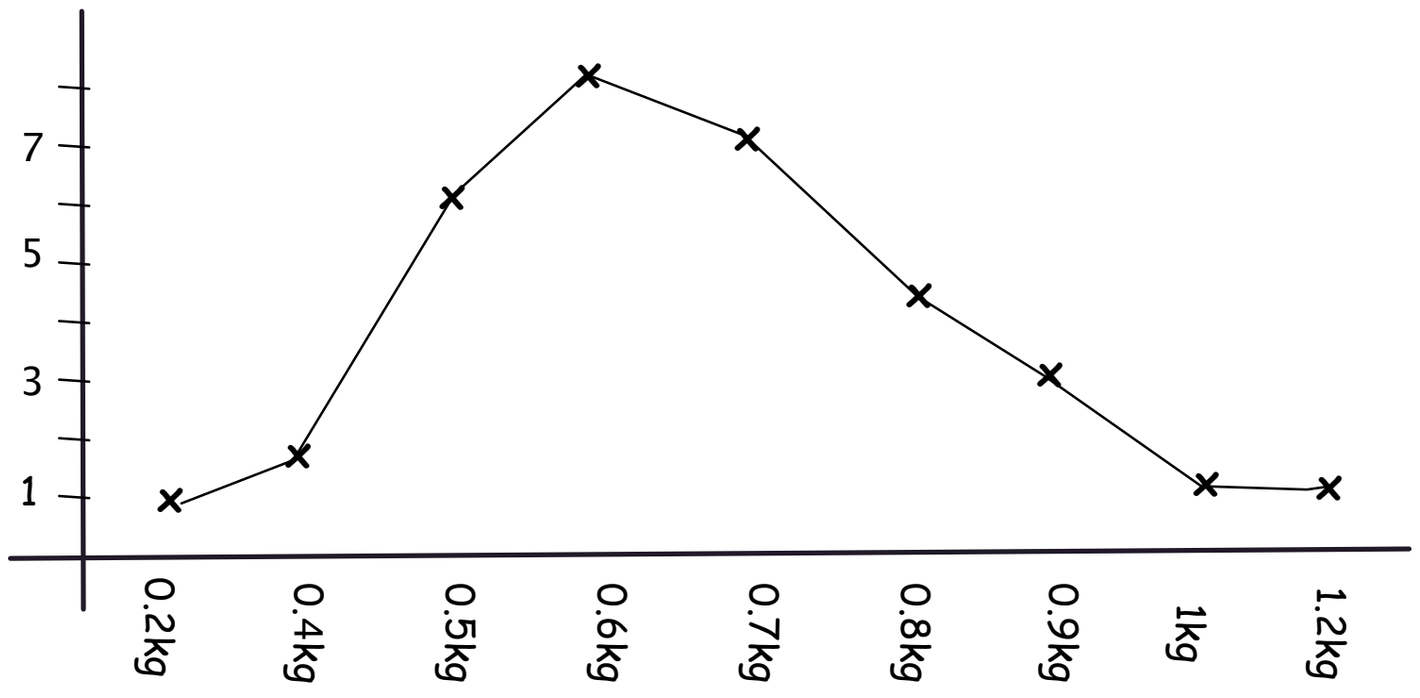
Statistics Definitions

Control, dependent and **independent variable**



First insight on the data

Weight	0.2kg	0.4kg	0.5kg	0.6kg	0.7kg	0.8kg	0.9kg	1kg	1.2kg
Number of chickens	1	2	6	8	7	4	3	1	1



Median and Mode (for discrete data)

{1, 1, 1, 2, 4, 4, 6, 7, 8}

median

{1, 2, 3, 4, 5, 6, 7, 8, 85, 88}

median

{1, 1, 1, 2, 4, 4, 6, 7, 8} mode = 1

Mode is the data that has the **highest frequency** in the set.

Median and Mode (for discrete data)

{1, 1, 1, 2, 4, 4, 6, 7, 8}

Multimodal means a dataset that has more than one peak in its distribution, which is different than multivariate.

Multivariate is a dataset that it has more than one information sources.

{1, 1, 1, 2, 4, 4, 6, 7, 8} mode = 1

Mode is the data that has the **highest frequency** in the set.

Variance, Standard Deviation and Covariance

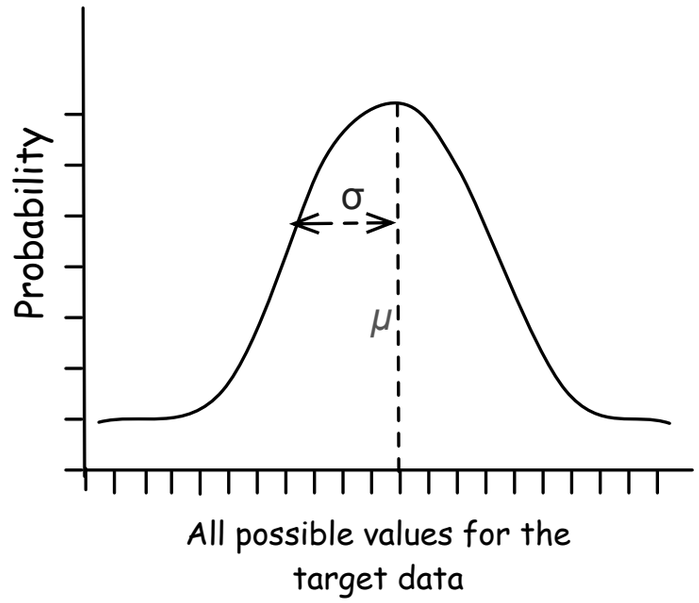
A single data point

Mean

Variance

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

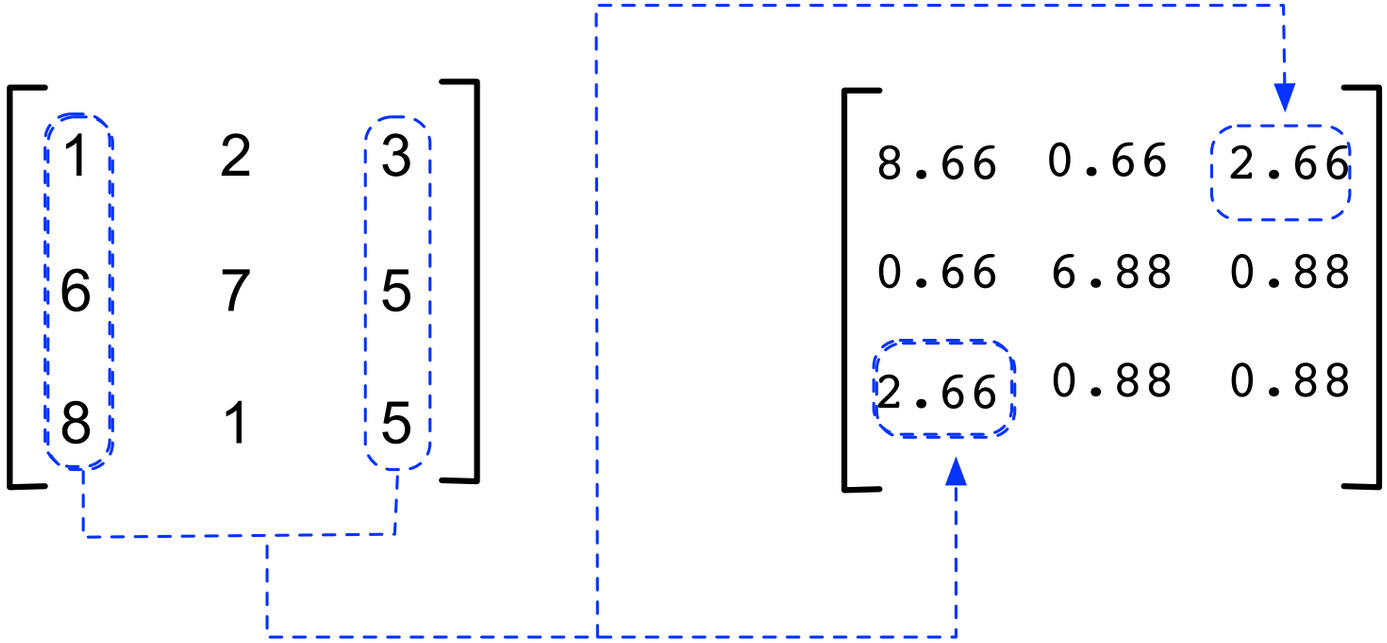
Number of data points



$$cov(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{n}$$

Covariance

The covariance between column 1 and column 3, is presented in the covariance matrix at cells 3,1 and cell 1,3, i.e., 2.66. Respectively, the covariance between column 1 and column 2, is presented in the covariance matrix at cells 1,2 and cell 2.1, i.e. 0.66



Python Code for Covariance

```
#------ Calculate covariance for a three variable dataset  ----------

import numpy as np

A = np.array([[1, 2, 3], [6, 7, 5], [8,1,5]])

np.cov(A, rowvar=False, bias=True)
```

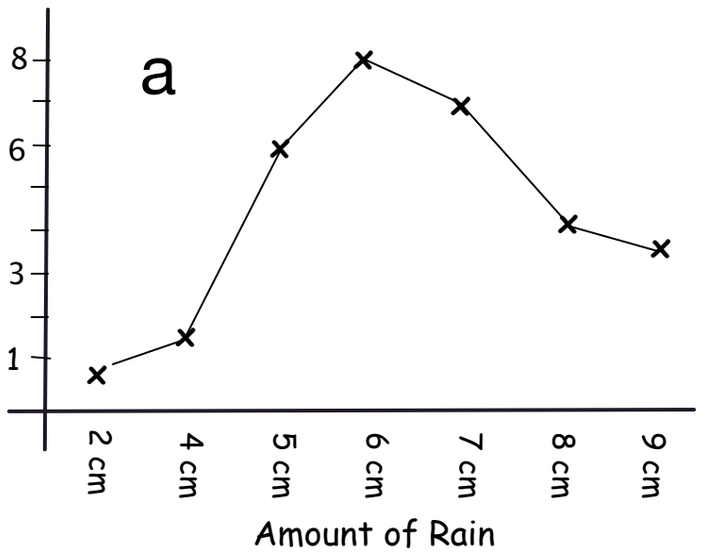
PDF - PMF - CDF

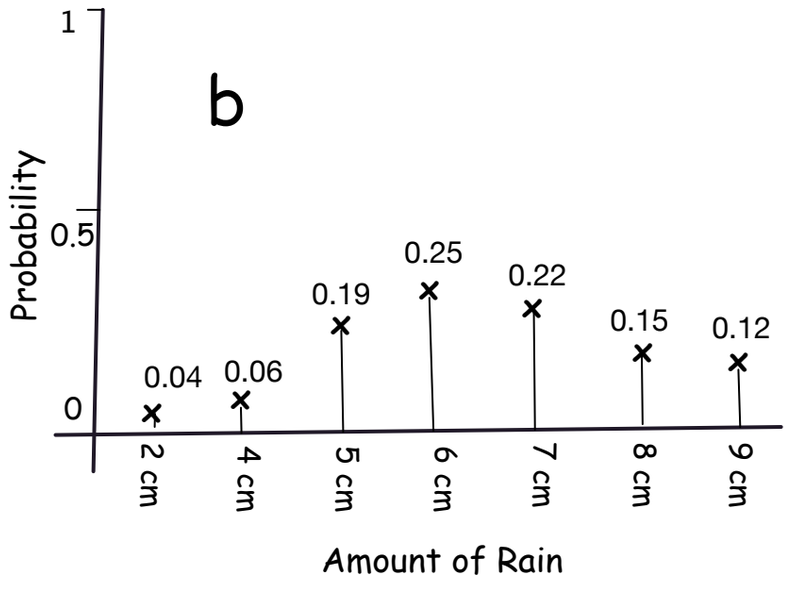
- Probability Density Function (PDF) shows how **dense** is the probability at **each data point** in a **continuous vector.**

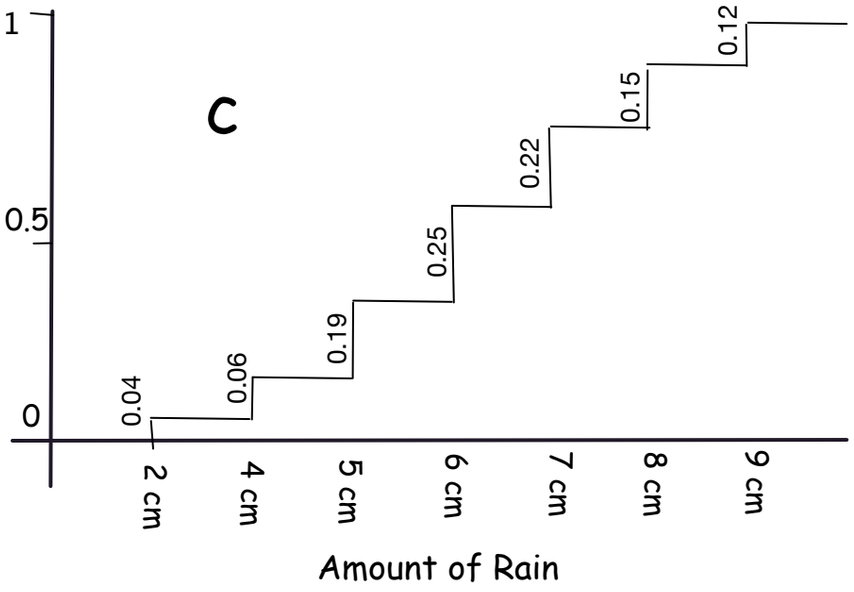
- Probability Mass Function (PMF) shows how **dense** is the probability at **each data point** in a **discrete vector**.

- Cumulative Distribution Function CDF is a function that describes a **distribution** of a variable (either discrete or continuous variable).

Example







distribution of rains

in r: dnorm

PMF

in r: pnorm

CDF

in r: qnorm

PDF - PMF - CDF

- Probability Density Function (PDF) shows how **dense** is the probability at **each data point** in a **continuous vector.**

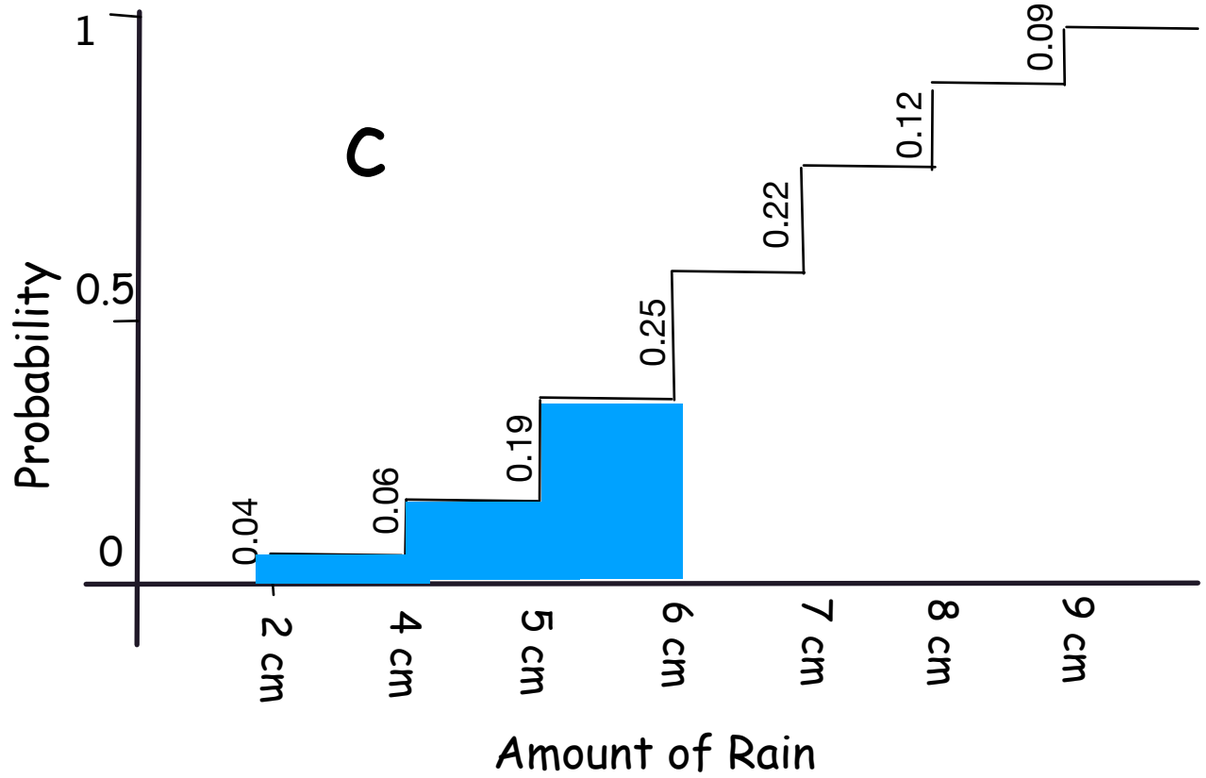
- Probability Mass Function (PMF) shows how **dense** is the probability at **each data point** in a **discrete vector**.

- Cumulative Distribution Function CDF is a function that describes a **distribution** of a variable (either discrete or continuous variable).

CDF is the probability of being $\leq x$ (x is a value), CDF is the cumulative PDF or PMF.

Example

What is the probability that raining is going to be less than 6cm or P ($x < 6$cm)?



Plotting PDF/PMF and CDF

```
#------ Plot sample PDF/PMF and CDF -------
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics

# Plot between -100 and 100 with 0.1 steps.
gaus_data = np.arange(-100, 100, 0.1)

# Calculating mean and standard deviation
mean = statistics.mean(gaus_data)
sd = statistics.stdev(gaus_data)

#---- plot PDF -----
pdf = norm.pdf(gaus_data, mean, sd)
plt.plot(gaus_data, pdf, label="PDF")
plt.show()

#---- plot CDF -----
cdf = np.cumsum(pdf)

plt.plot( cdf, label="CDF")
plt.legend()
plt.show()
```

Statistical Distributions

Distributions

- A distribution presents characteristics of a dataset (descriptive statistics)

- Remember: *Any data object must include repetitive values in a dataset, because any scientific phenomena should be reproducible.*

- The distribution presents all possible values of a single variable or **how often they occur** (probability or frequency of their occurrences).

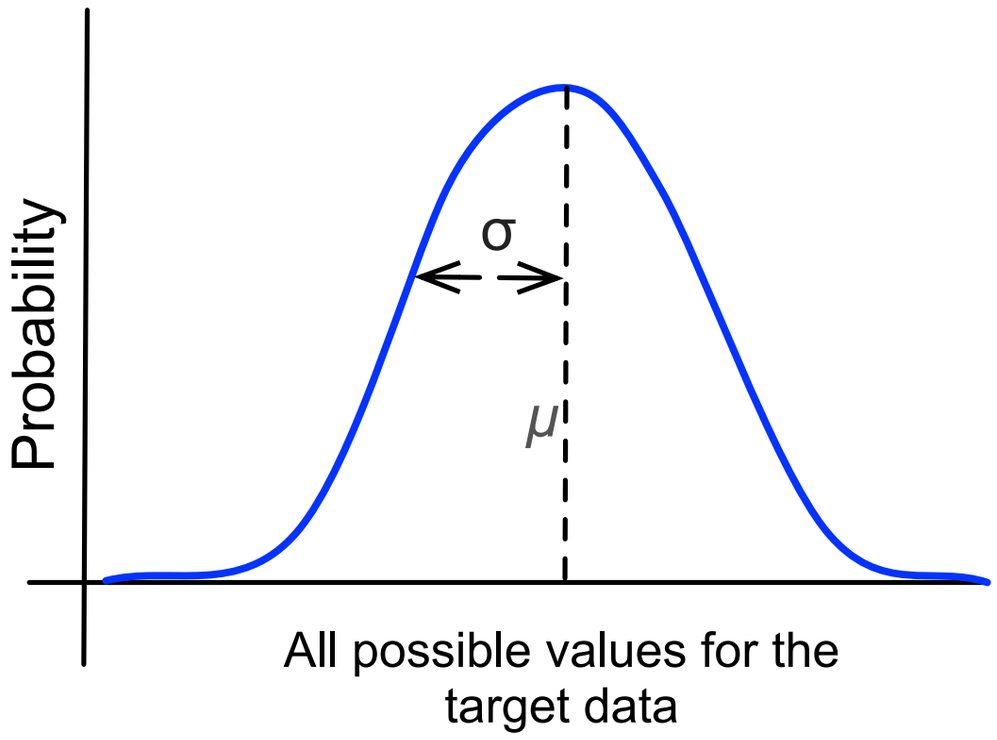
Distributions

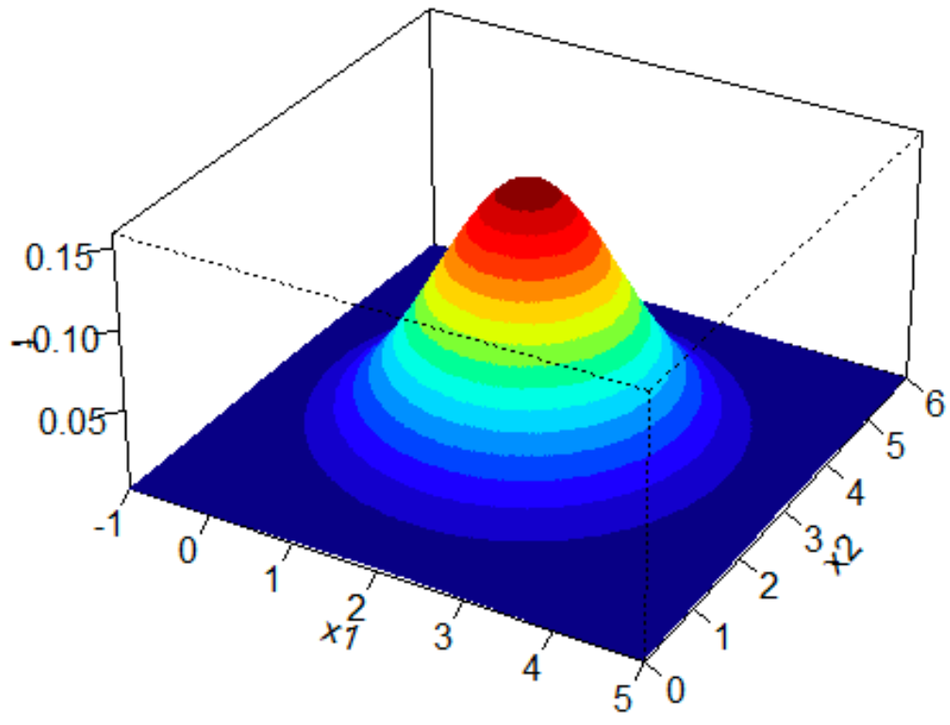
- A distribution presents characteristics of a dataset (descriptive statistics)
- Remem*ues in a dataset, reproduc*
- The dist*ngle* variable or **how often they occur** (probability or frequency of their occurrences).

1) Usually we use distribution to plot a variable in a two dimensional space, i.e. X axis presents **values** and Y presents the **volumes**.

2) Usually distribution only focuses on **one single data point (variable).** Otherwise, it is called multidimensional distribution

Normal (Gaussian) Distribution





Gaussian Distribution

The following equation presents the PDF of Gaussian distribution for any given variables of *x*. However, you do not need to memorize it.

$$f(x) = \frac{1}{\sigma \times \sqrt{2\pi}} \times e^{-\frac{(x-\mu)^2}{2 \times \sigma^2}}$$

If you can't recall from your school, both Pi, $\pi = 3.14$, and Euler, $e = 2.71$, are constant numbers in mathematics, and we simply can substitute them.

A multivariate normal distribution is a covariance matrix instead of variance. For example, a two-dimensional normal distribution uses the following vector for the mean and covariance matrix (Σ), assuming ρ a correlation between two dimensions.

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \sum = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

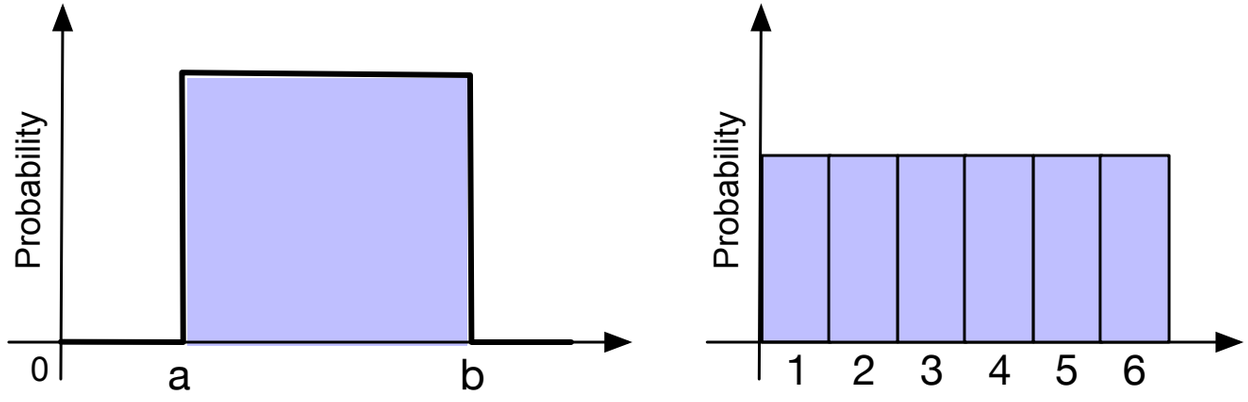
Uniform Distribution

- A uniform distribution (rectangular distribution), is a distribution in which all its outcomes are equally likely (they have the same probability).

- Examples:
  - Throwing dice: All chances have equal probability; we have 6 options, and the chance to get a particular number is $1/6$.
  - Flipping a coin: We can get either a head or tail, and both have an equal probability, which is $1/2$.
  - The above two examples are discrete uniform distributions. Uniform distribution could also be from continuous data. For example, an algorithm that generates random numbers has a continuous uniform distribution.

Uniform Distribution

- The result of trials lay between certain bounds in this distribution, and these bounds are defined as parameters *a* (minimum) and *b* (maximum), and the Uniform distribution is written as $U(a, b)$.

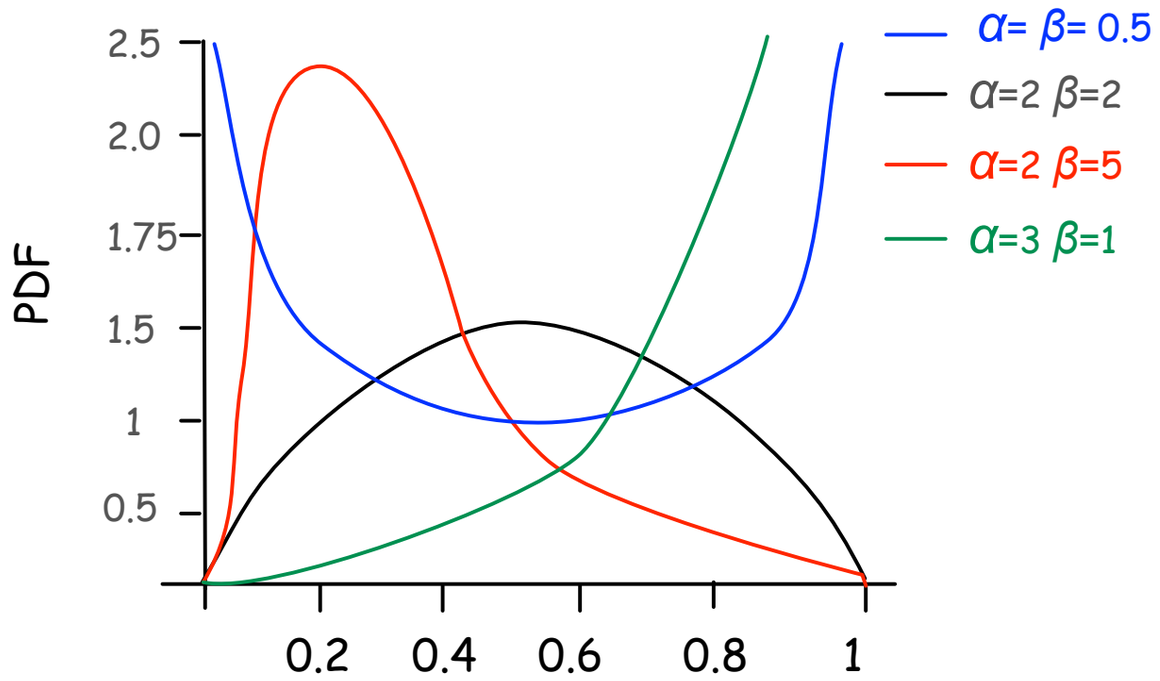


(left) Continuous Uniform distribution (right) dice tossing distribution, which is uniform and discrete

The PDF of continuous Uniform distribution is written as follows:

$$f(x) = \begin{cases} \frac{1}{a-b} & for \quad a \le x \le b \\ 0 & for \quad x < a \ or \ x > b \end{cases}$$

Beta Distribution



The Beta distribution is suitable for the **random behavior of a binary trail** (yes/no, success/failure, coin head/tail,…). In other words, when there is an uncertainty in a binary probability result we can use Beta distribution to understand the **conditional distribution** of a success rate (success is a binary state such as true/ok/yes…).

This distribution is useful in scenarios where *the exact probability of a binary outcome (such as success/failure or yes/no) is uncertain.*
The Beta distribution allows for a flexible way to express and manage this uncertainty, accommodating a range of beliefs about what the true probability might be.

Beta Distribution

- Beta distribution is a distribution that is parameterized by θ given α and β parameters, and its PDF is written as follows:

$$P(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

θ is in the range between 0 and 1, $\theta \in [0,1]$. B is the normalization constant that ensures the total probability is 1. Therefore, we can take out $B(\alpha, \beta)$ and say that the $P(\theta \mid \alpha, \beta)$ is proportional to $\theta^{\alpha-1}(1-\theta)^{\beta-1}$.

Now, by substituting values for α and β in $\theta^{\alpha-1}(1-\theta)^{\beta-1}$, we get these distribution shapes that have been shown in the presented Figure.

Beta Distribution

- Beta distribution is a distribution that is parameterized by θ given α and β parameters, and its PDF is written as follows:

$$P(\theta \mid \alpha, \beta) = \frac{}{}$$

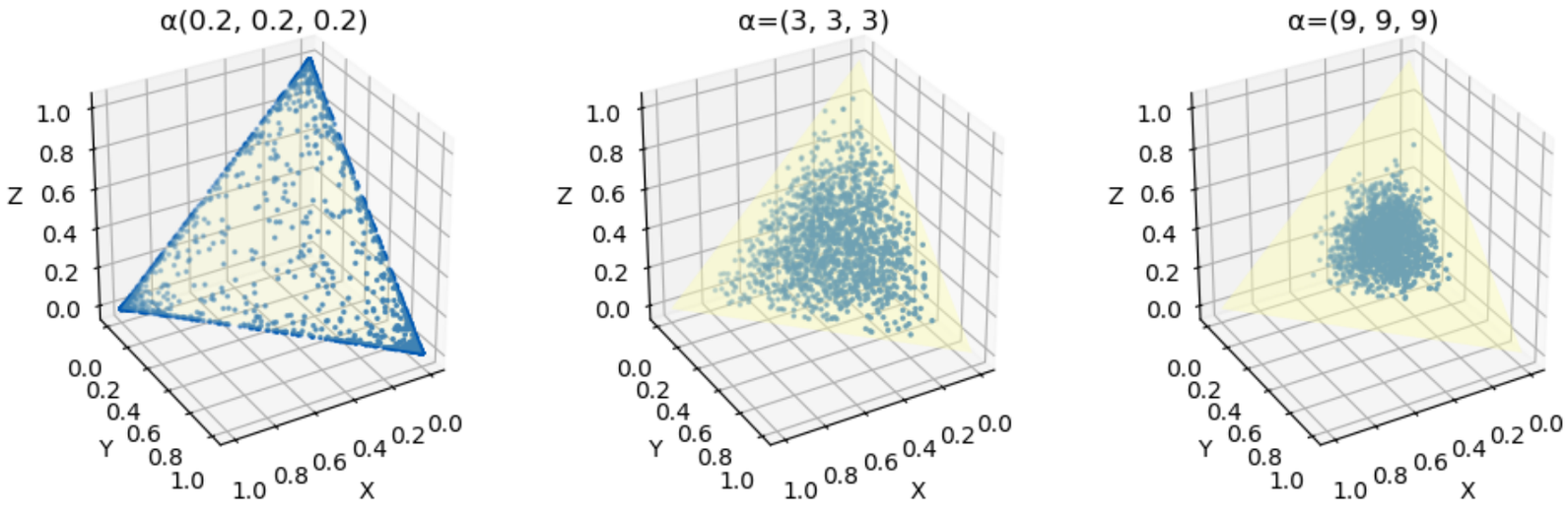
https://en.wikipedia.org/wiki/Beta_distribution#/media/File:PDF_of_the_Beta_distribution.gif

θ is in the range b e normalization constant that ensure we can take out $B(\alpha, \beta)$ and say that $(1-\theta)^{\beta-1}$.

Now, by substituting values for α and β in $\theta^{\alpha-1}(1-\theta)^{\beta-1}$, we get these distribution shapes that have been shown in Figure.

Dirichlet Distribution

- Similar to Beta distribution but for multiple variables instead of one variable.



α is a vector of non-zero numbers.

k is the dimension number of the vector. In this example k=3.

Dirichlet Distribution

Gamma function (Euler Gamma Function). Gamma function behaves like a factorial for natural numbers but generalizes it to positive real numbers (a continuous set).

$$\Gamma(n) = \int_0^\infty x^{n-1} e^{-x} dx \quad , \quad n > 0$$

PDF of Dirichlet Distribution:

$$f(\theta_1, \theta_2, \ldots, \theta_m) = \frac{\Gamma(A)}{\prod_{i=1}^{m} \Gamma(a_i)} \prod_{i=1}^{m} \theta_i^{a_i - 1}$$

Binomial Distribution

There are cases that our variable is discrete and it has only two discrete states (binary), such as flipping a coin (head or tail), the door state (open or close) or whether you find this course helpful (yes, no). In these cases we use **binomial distribution**

Binomial Distribution

There are cases that our variable is discrete and has only two discrete states (binary), such as flipping a coin (head or tail), the door state (open or close) or whether you find this course helpful (yes, no). In these cases we use **binomial distribution**

Independent trail: each trial (coin flipping event) does “not” have any connection to the previous trial, e.g. flipping coin, dice,…

Dependent trail: the outcome of a trail depends on its previous trails, e.g. taking anti depressant pills. (1st is good, 2nd is great, 3rd sleepy, 4th more depressed)

Binomial Distribution

The binomial distribution is appropriate if three conditions are all true.

(i) We have a series of **independent trials**.

(ii) Trail **output is binary**, and denoted as success or failure, but it could be other information as well such as yes/no, true/false, etc.

(iii) **There is a finite number of trails**.

Binomial coefficient

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

The binomial PDF for a given value *x* and given pair of parameters *n, p* and *q* is written as follows:

$$f(x \mid n, p) = \binom{n}{x} p^x q^{(n-x)}$$

$f(x \mid n, p)$ presents the probability of observing exactly *x* successes in *n* independent trials, where the probability of success in any given trial is *p* and the probability of failure in any given trial is *q.*

One of the eight following situation could happen for these three chickens:

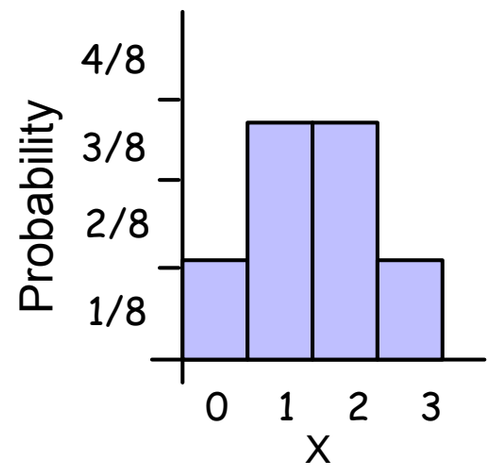
{h,h,h}, {h,h,n}, {h,n,n}, {n,n,n},
{n,h,h}, {n,h,n}, {n,n,h}, {h,n,h}.

h: heart attack
n: not heart attack

$$P(\textit{no heart attack}) = P(\{n,n,n\}) = 1/8$$
$$P(\{\textit{one gets heart attack}\}) = P(\{h,n,n\}) + P(\{n,h,n\}) + P(\{n,n,h\}) = 1/8 + 1/8 + 1/8 = 3/8$$
$$P(\{\textit{two get heart attack}\}) = P(\{h,h,n\}) + P(\{n,h,h\}) + P(\{h,n,h\}) = 1/8 + 1/8 + 1/8 = 3/8$$
$$P(\{\textit{all three get heart attack}\}) = P(\{h,h,h\}) = 1/8$$

By using x as a variable that specifies the number of chickens getting heart-attack, we can rewrite the above probabilities as follows:

$P(\textit{zero heart attack}) = P(x = 0)$
$P(\textit{one heart attack}) = P(x = 1)$
$P(\textit{two heart attack}) = P(x = 2)$
$P(\textit{three heart attack}) = P(x = 3)$



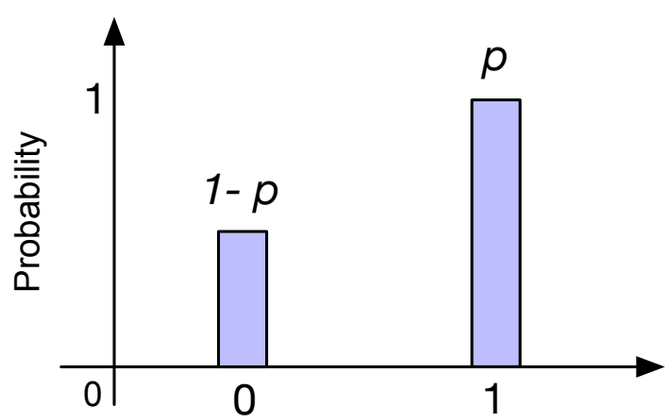
Bernoulli Distribution

An experiment whose result is random and can only have one binary outcome is known as a Bernoulli trial. For instance, we have one chicken, and this chicken has eaten too much; either it gets a heart attack or does not get a heart attack. In this case, the probability of a heart attack is $\frac{1}{2}$, and the probability of not getting a heart attack is $1 - \frac{1}{2} = \frac{1}{2}$.

Bernoulli distribution is used for discrete data,

PMF of Bernoulli distribution:

$$P(n) = \begin{cases} 1 - p & for \quad n = 0 \\ p & for \quad n = 1 \end{cases}$$



Geometric Distribution

(i) there is a series of **independent** trails.

(ii) trail's **output is binary,** e.g. success/failure, yes/no, true/false, etc.

(iii) As the desired binary state is acquired the **trial stops immediately.**

Geometric distribution is a specific case of Binomial distribution, it is discrete and has **only one trail.**

Considering *p* provides the probability of success, *x* is the number of trials ($x = 1,2,...$), The PMF of Geometric distribution is written as: $f(x) = p(1-p)^x$

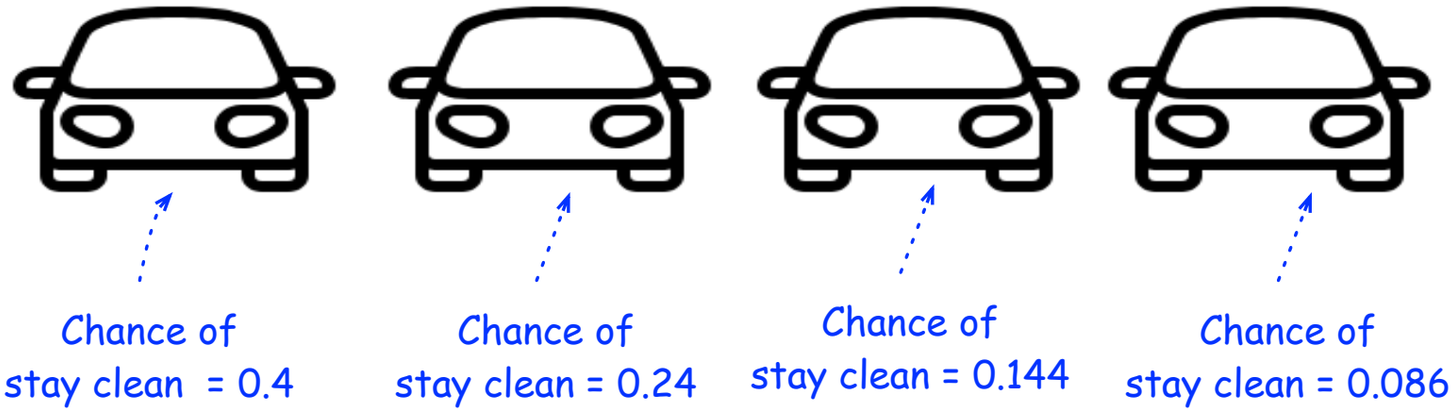
The probability that the pigeon makes a clean window dirty is **P=0.6**, so not making it dirty is **1-0.6 = 0.4**. X denotes the number of cars he passes without making them dirty. Assuming, “not making dirty” = “success” and “making dirty” = “failure”.

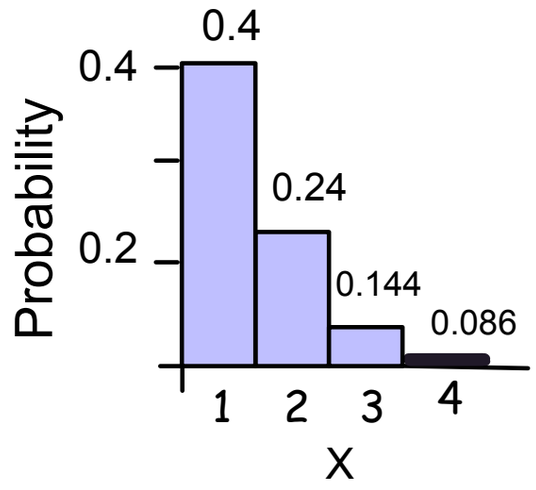
$$P(X=1) = P(success\ in\ the\ 1^{st}\ trial) = 0.4$$

$$P(X=2) = P(failure\ in\ the\ 1^{st}\ trial) \times P(success\ i\ nthe\ 2^{nd}\ trial) = 0.6x0.4 = 0.24$$

$$P(X=3) = P(failure\ in\ the\ 1^{st}\ trial) \times P(failure\ in\ the\ 2^{nd}\ trial) \times$$
$$P(success\ in\ the\ 3^{rd}\ trial) = 0.6x0.6x0.4 = 0.144$$

$$P(X=4) = P(failure\ in\ the\ 1^{st}\ trial) \times P(failure\ in\ the\ 2^{nd}\ trial) \times$$
$$\times P(failure\ in\ the\ 3^{rd}\ trial) \times P(success\ in\ the\ 3^{rd}\ trial) = 0.6 \times 0.6 \times 0.6 \times 0.4$$
$$= 0.086$$





Poisson Distribution

- There are **rare** events happening in a system (and they exist in the dataset), such as malfunctions of a machine. We know the average occurrences of these rare events (in time) and their frequency is not changing.
- Poisson distribution is being used to model the **intervals** of **rare events**.

The **value of mean and variance are equal and it is shown as λ (lambda).** Don't forget that we said that the **average (mean) is not changing**.

'e' is a mathematical constant similar to **π** and it is **2.718**

Mean

$$P(X = r) = \frac{e^{-\lambda} \cdot \lambda^{r}}{r!}$$

The number of rare event occurrences

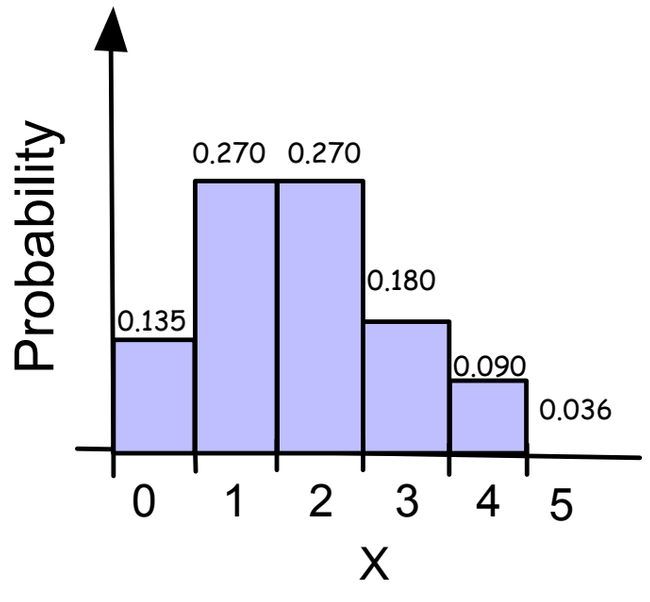
We have a rare event and the mean occurrences of this event per month is 2, λ=2 x 4 (month) = 8, in other words the mean of four months is 8.

$$P(X=3)=\frac{e^{-8}.8^{3}}{3!}=0.286$$

What is the probability that we get zero failure in a month (per month λ=2)?

$$P(X=0)=\frac{e^{-2}.2^{0}}{0!}=\frac{e^{-2}.1}{1}=0.135$$

$$P(X=1)=\frac{e^{-2}.2^{1}}{1!}=0.270$$



Weibull Distribution

If the rare events we mentioned occur at a constant rate Poisson distribution is appropriate.

Nevertheless, if they do not occur at a constant rate and time, we cannot identify their rate; we can use the *Weibull distribution*.

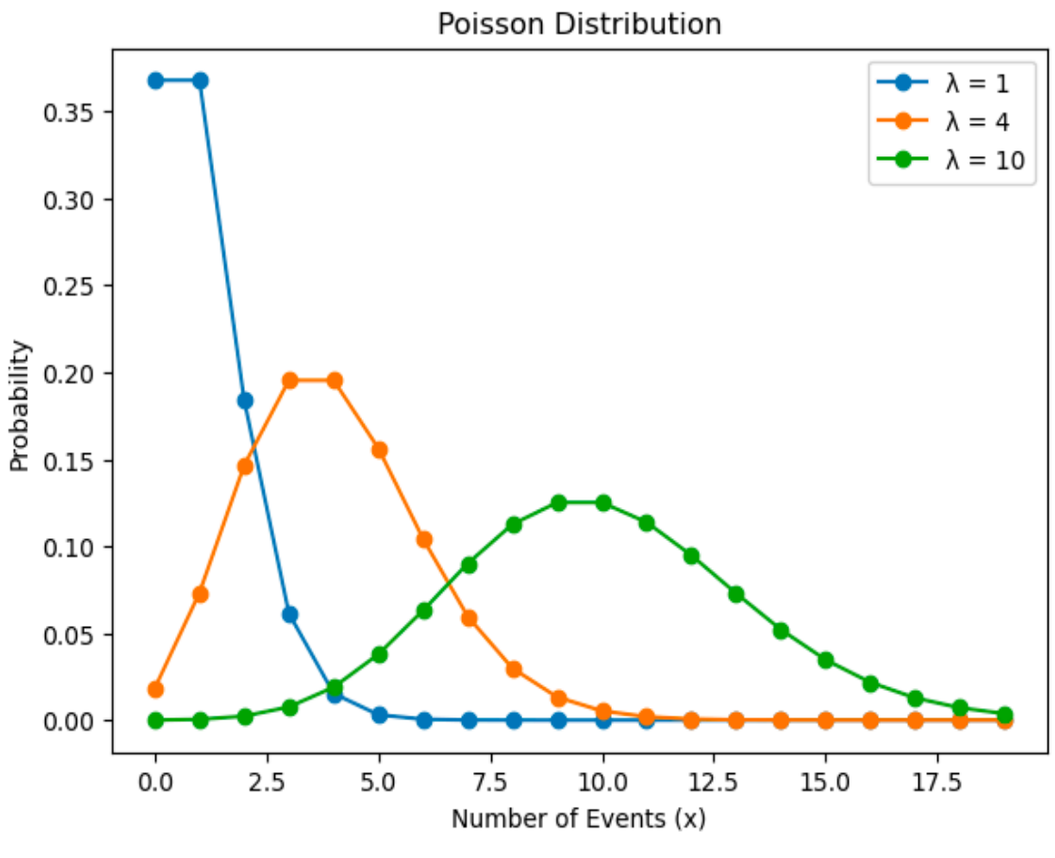
Weibull distribution models a distribution where the rate of rare events is not constant and it may vary over time. The shape of a Weibull distribution depends on a parameter k, which is known as the shape parameter.

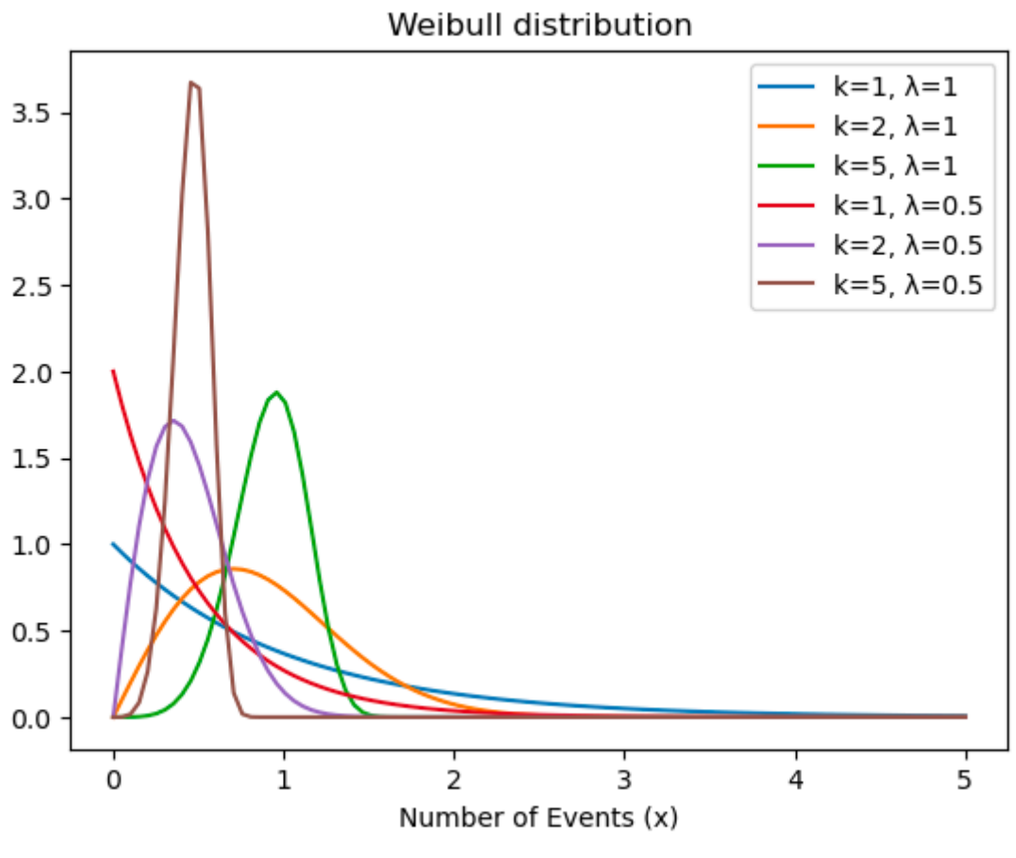
Weibull Distribution

The Weibull distribution is a continuous probability distribution. Thus, it is characterized by a Probability Density Function (PDF) rather than a Probability Mass Function (PMF). Its PDF is written as follows:

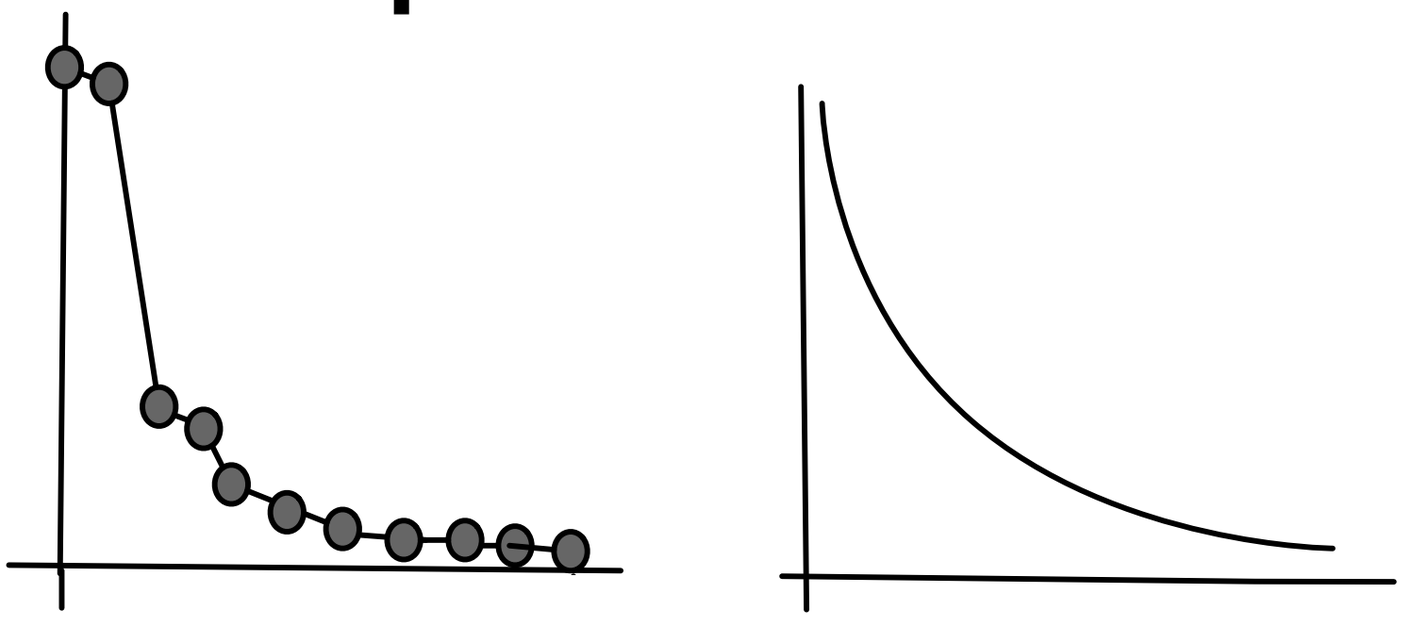
$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & \text{for } x \geq 0, \\ 0 & \text{for } x < 0. \end{cases}$$

Here, x is the variable (the value for which we are calculating the PDF), lambda is the scale parameter and k is the shape factor.





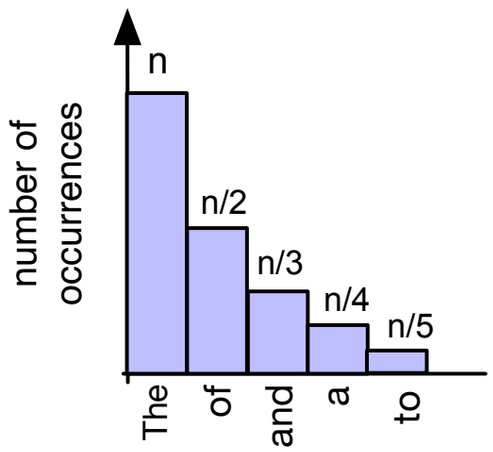
Power Law (Long Tail), Exponential, and ZipfLaw Distributions



1

PDF: $f(x)= x^{\alpha}$. ’α’ is called power law exponent and causes this exponential changes, it is constant.

Zipflaw

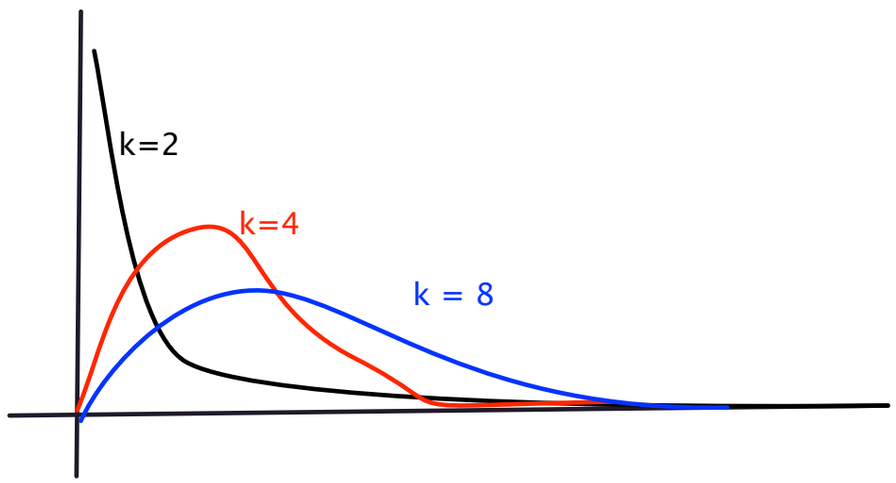


Experiment with Paper Clips

Chi-Squire Distribution

The chi-square distribution is non-symmetrical, skewed to the right side of the X-axis, in which the shape of the distribution is very much dependent on the degree of freedom k. The mean in the chi-square distribution is equal to the degree of freedom; as the degree of freedom increases, this distribution is skewed toward a normal distribution.

This distribution is being used for the chi-square test, including the test for **Goodness-of-Fit** and **dependence between two categorical variables**; we explain both of these uses later. Besides, it is used to test whether a sample dataset variance equals a hypothesized population variance and estimate confidence intervals for given variance and standard deviation.



Chi-square distribution with three different degree of freedom.

Chi-Square Distribution

PDF of Chi-square distribution is written as follows:

$$f(x,k) = \frac{x^{k-1}e^{-x^2/2}}{2^{(k/2)-1}\Gamma(\frac{k}{2})} \quad for \quad x \geq 0 \text{ for } x \geq 0$$

Γ is the Gamma function, *k* refers to the degree of freedom or independent (input) variables that have normal distributions.

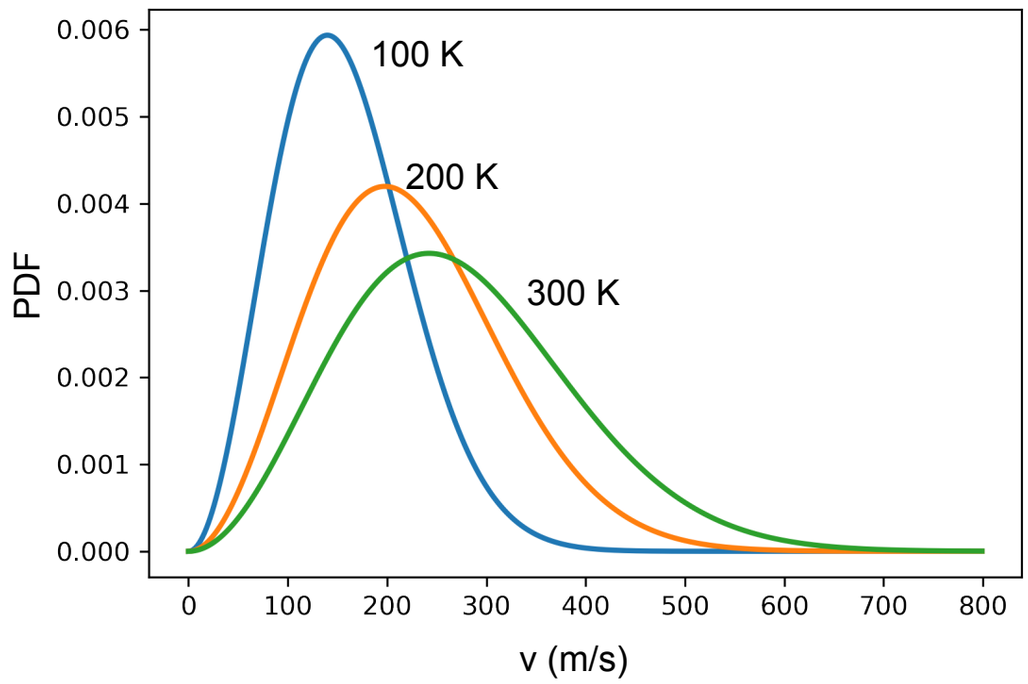
Boltzmann Distribution

- Boltzmann (Maxwell-Boltzmann or Gibbs) distribution is used to model the following statement: *the energy gets distributed from high density to places that have lower density until there is a balance between energy distribution density (thermal equilibrium)*.

- We have learned in school that when we increase the heat (energy), gas molecules are starting to move faster and their kinetic energy increases. => *the temperature is proportional to average kinetic energy.*

Boltzmann Distribution

- For example, consider we have three containers (A, B, and C) of a gas molecule. Container A's temperature is 300 Kelvin, container B's temperature is 200 Kelvin, and container C's temperature is 100 Kelvin. The average kinetic energy of molecules in container A is higher than the average kinetic energy of molecules in container B, and container B's average kinetic energy of molecules is higher than container C. This means molecules in this container are moving faster (higher velocity).



Boltzmann Distribution (Thermal Equilibrium)

- All systems tend to move toward thermal equilibrium. “Thermal equilibrium” refers to a condition when parameters do not exchange any energy (the high energy moves to low energy until there is a balance in energy everywhere).

- Low energy means high probability in that state, and high energy means low probability in that state.

- To understand Thermal equilibrium considers that we connect all containers A, B and C together. Their temperature will be changed to something like the average temperature in each.

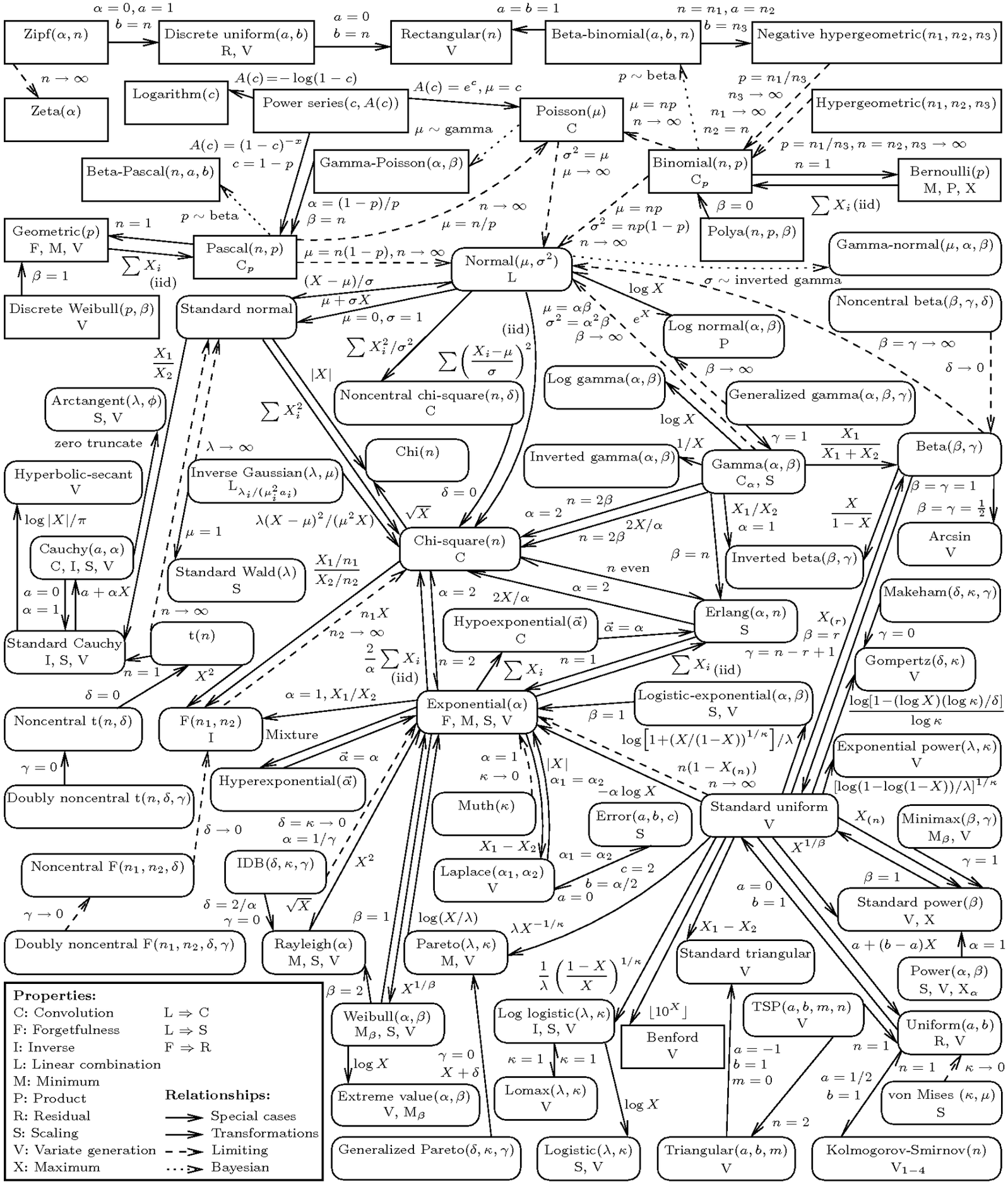
Boltzmann Distribution

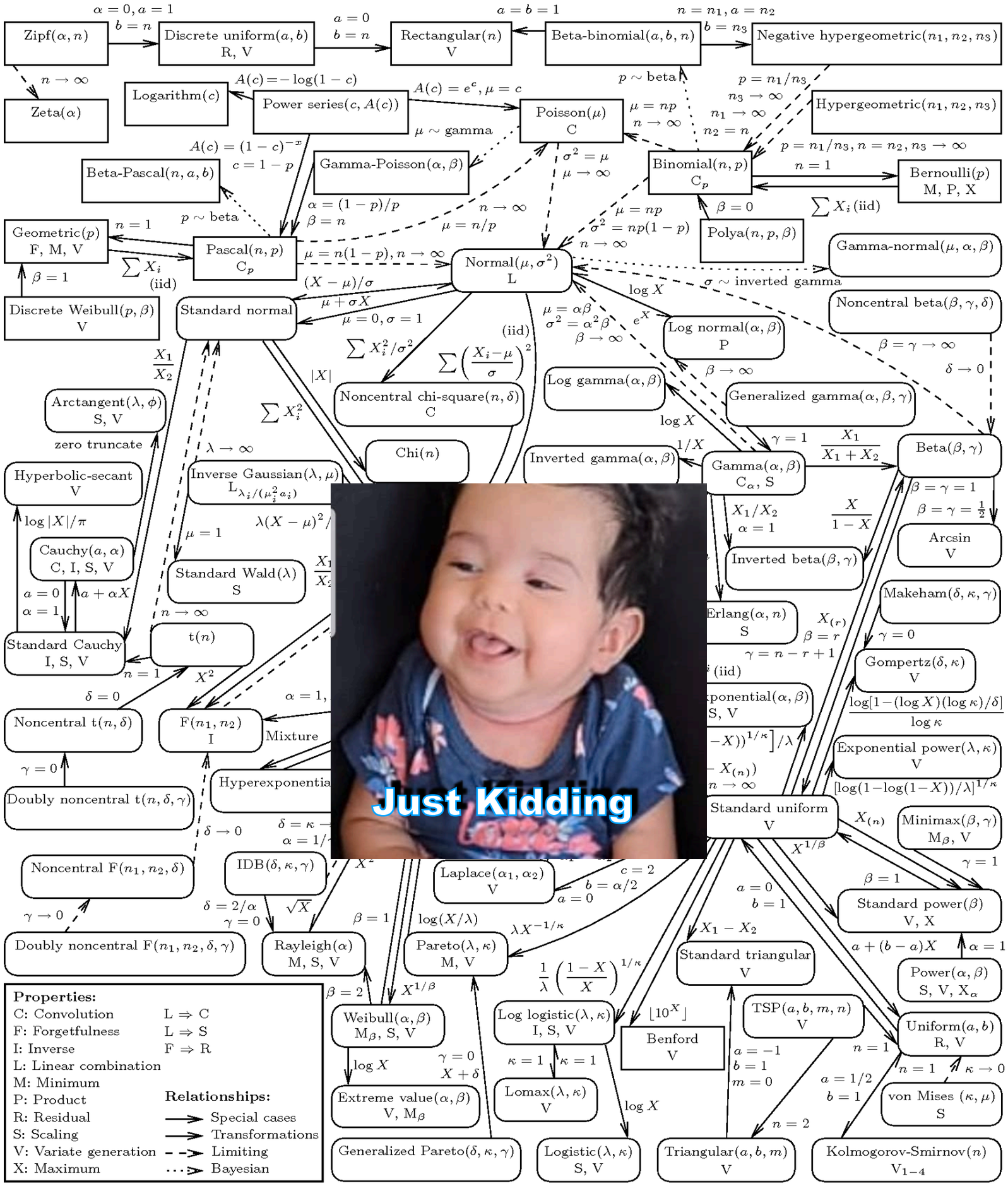
- In the context of machine learning the Boltzmann (Maxwell-Boltzmann or Gibbs) distribution represents the probability for the distribution of the states in a system, based on the different energy levels in the system. The PDF of Boltzmann distribution is as follows:

$$p_i = \frac{e^{-\epsilon_i/kT}}{\sum_{j=1}^{M} e^{-\epsilon_j/kT}}$$

- In this equation, p_i is the probability of state i, ϵ_i presents the energy at state *i*, k is the Boltzmann constant and T is the temperature of the system.

- Keep in mind that Boltzmann distribution is based on *the probability of a state in the system, is inversely related to the energy of the system at that state*.


Zipf(α, n)
Discrete uniform(a, b)
R, V
Rectangular(n)
V
Beta-binomial(a, b, n)
Negative hypergeometric(n1, n2, n3)
Zeta(α)
Logarithm(c)
Power series(c, A(c))
Poisson(μ)
C
Hypergeometric(n1, n2, n3)
Beta-Pascal(n, a, b)
Gamma-Poisson(α, β)
Binomial(n, p)
Cp
Bernoulli(p)
M, P, X
Geometric(p)
F, M, V
Pascal(n, p)
Cp
Polya(n, p, β)
Gamma-normal(μ, α, β)
Normal(μ, σ2)
L
Discrete Weibull(p, β)
V
Standard normal
Noncentral beta(β, γ, δ)
Log normal(α, β)
P
Log gamma(α, β)
Generalized gamma(α, β, γ)
Noncentral chi-square(n, δ)
C
Arctangent(α, φ)
S, V
Chi(n)
Inverted gamma(α, β)
Gamma(α, β)
Cα, S
Beta(β, γ)
Hyperbolic-secant
V
Inverse Gaussian(λ, μ)
Lλi/(μi2ai)
Chi-square(n)
C
Arcsin
V
Cauchy(a, α)
C, I, S, V
Standard Wald(λ)
S
Inverted beta(β, γ)
Makeham(δ, κ, γ)
Hypoexponential(α⃗)
C
Erlang(α, n)
S
Gompertz(δ, κ)
V
Standard Cauchy
I, S, V
t(n)
Exponential(α)
F, M, S, V
Logistic-exponential(α, β)
S, V
Exponential power(λ, κ)
V
Noncentral t(n, δ)
F(n1, n2)
I
Mixture
Hyperexponential(α⃗)
Muth(κ)
Error(a, b, c)
S
Standard uniform
V
Minimax(β, γ)
Mβ, V
Doubly noncentral t(n, δ, γ)
Noncentral F(n1, n2, δ)
IDB(δ, κ, γ)
Laplace(α1, α2)
V
Standard power(β)
V, X
Doubly noncentral F(n1, n2, δ, γ)
Rayleigh(α)
M, S, V
Pareto(λ, κ)
M, V
Standard triangular
V
Power(α, β)
S, V, Xα
Weibull(α, β)
Mβ, S, V
Log logistic(λ, κ)
I, S, V
TSP(a, b, m, n)
V
Benford
V
Uniform(a, b)
R, V
Extreme value(α, β)
V, Mβ
Lomax(λ, κ)
V
von Mises (κ, μ)
S
Generalized Pareto(δ, κ, γ)
Logistic(λ, κ)
S, V
Triangular(a, b, m)
V
Kolmogorov-Smirnov(n)
V1−4
Properties:
C: Convolution
F: Forgetfulness
I: Inverse
L: Linear combination
M: Minimum
P: Product
R: Residual
S: Scaling
V: Variate generation
X: Maximum
L ⇒ C
L ⇒ S
F ⇒ R
Relationships:
Special cases
Transformations
Limiting
Bayesian


Zipf(α, n)
Discrete uniform(a, b)
R, V
Rectangular(n)
V
Beta-binomial(a, b, n)
Negative hypergeometric(n1, n2, n3)
Zeta(α)
Logarithm(c)
Power series(c, A(c))
Poisson(μ)
C
Hypergeometric(n1, n2, n3)
Beta-Pascal(n, a, b)
Gamma-Poisson(α, β)
Binomial(n, p)
Cp
Bernoulli(p)
M, P, X
Geometric(p)
F, M, V
Pascal(n, p)
Cp
Polya(n, p, β)
Gamma-normal(μ, α, β)
Normal(μ, σ2)
L
Discrete Weibull(p, β)
V
Standard normal
Noncentral beta(β, γ, δ)
Log normal(α, β)
P
Log gamma(α, β)
Generalized gamma(α, β, γ)
Noncentral chi-square(n, δ)
C
Chi(n)
Inverted gamma(α, β)
Gamma(α, β)
Cα, S
Beta(β, γ)
Arctangent(λ, φ)
S, V
Hyperbolic-secant
V
Inverse Gaussian(λ, μ)
Cauchy(a, α)
C, I, S, V
Standard Wald(λ)
S
Arcsin
V
Inverted beta(β, γ)
Makeham(δ, κ, γ)
Erlang(α, n)
S
Standard Cauchy
I, S, V
t(n)
Gompertz(δ, κ)
V
Noncentral t(n, δ)
F(n1, n2)
I
Mixture
Exponential(α, β)
S, V
Exponential power(λ, κ)
V
Doubly noncentral t(n, δ, γ)
Standard uniform
V
Minimax(β, γ)
Mβ, V
Noncentral F(n1, n2, δ)
IDB(δ, κ, γ)
Laplace(α1, α2)
V
Standard power(β)
V, X
Doubly noncentral F(n1, n2, δ, γ)
Rayleigh(α)
M, S, V
Pareto(λ, κ)
M, V
Standard triangular
V
Power(α, β)
S, V, Xα
Weibull(α, β)
Mβ, S, V
Log logistic(λ, κ)
I, S, V
Benford
V
TSP(a, b, m, n)
V
Uniform(a, b)
R, V
Extreme value(α, β)
V, Mβ
Lomax(λ, κ)
V
von Mises (κ, μ)
S
Generalized Pareto(δ, κ, γ)
Logistic(λ, κ)
S, V
Triangular(a, b, m)
V
Kolmogorov-Smirnov(n)
V1−4
Just Kidding
Properties:
C: Convolution
F: Forgetfulness
I: Inverse
L: Linear combination
M: Minimum
P: Product
R: Residual
S: Scaling
V: Variate generation
X: Maximum
L ⇒ C
L ⇒ S
F ⇒ R
Relationships:
Special cases
Transformations
Limiting
Bayesian

Summary

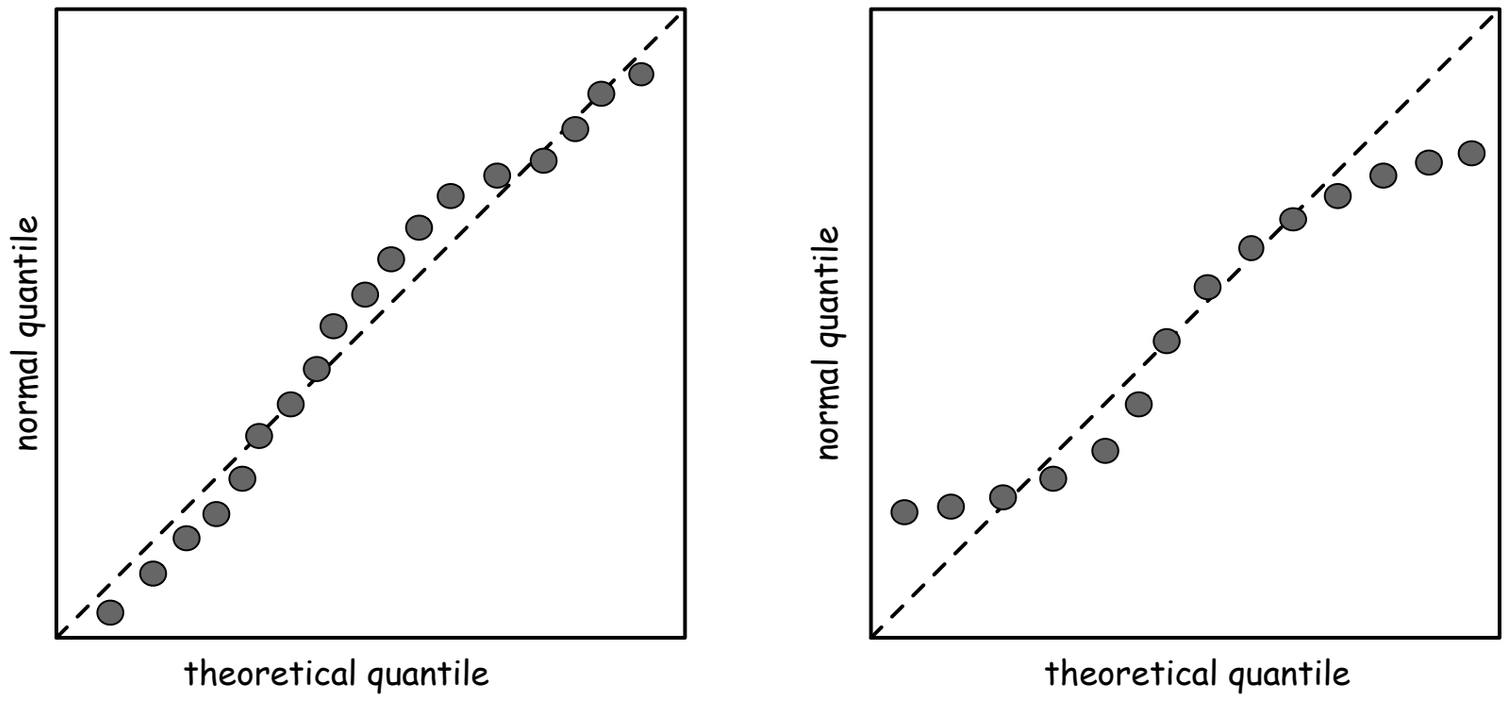
Normal	Uniform	Beta	Dirchilet
- bell curved - used to present wether we have collected enough data or not. - two of its well-known subtypes are z-distribution and t-distribution	- used when all outcomes of sample has equal probability. - Its discrete version has finite outcome and its a continuous version has an infinite outcome.	- used when there is an uncertainty in binary trail and we don't have information about their underlying probabilities. - defined by two parameters α and β	- multivariate generalization of Beta distribution - defined by two parameters α and β - use in topic modeling, i.e. latent dirchilet analysis

Binomial	Bernoulli	Geometric	Power Law / Exponential
- used when there is a series of binary independent trail. - it can be used for making inferences about the binary trails in terms of probability.	- A special type of Binomial distribution that has only one trial.	- very similar to Binomial, except that after the first encounter the trail stops. - it can be used for making inferences about the binary trails in terms of probability.	- observed in many real-world phenomena including physics, biology, literature,.. - it is characterized by a parameter called α. - Two subtype of this distribution are Zipf law and Pareto distribution.

Poisson	Weibull	Chi-Square	Boltzmann
- used to model a rare event that is happening in the system, in a particular interval. - it can be used for making inferences about the binary trails in terms of probability.	- used to model a rare event that is happening in the The system, in the different intervals. - it can be used for making inferences about the binary trails in terms of probability.	- used to test Goodness-of-Fit - used to test dependence between two categorical variables	- Describes the probability of a system being in a certain state as a function of that state's energy and the temperature of the system. - The energy gets distributed from high density to places with lower density until there is a balance between energy distribution density (thermal equilibrium).

How to check whether the distribution existed or not?

- We can use P-P plot, or Q-Q plot.
- Two Q-Q plots, the one on the left is close to a straight line, which means our dataset is following the desired distribution. The one on the right does not follow the desired distribution because it does not create a straight line.



Python Codes for Poisson Distributions

```
#----- Poisson distribution python -------

from scipy.stats import poisson

import numpy as np
import matplotlib.pyplot as plt
```

```
# rvs creates some random data with the given mean and loc

y = poisson.rvs(mu=2, size=100000) # ***TODO in the class***: play
with mu =2,6, 20,
# plotting the graph
plt.hist(y,density=True)

# showing the graph
plt.show()
```

Python code for Binomial Distribution

```
from numpy import random
from scipy.stats import binom

#generate an array of 10 values that follow a
binomial distribution
random.binomial(n=10, p=.5, size=10)


#calculate binomial probability. The probability
of success is 60%, if the number of trials are 12,
#what is the probability of having 10 successful.
binom.pmf(k=10, n=12, p=0.6)
```

Python Codes for Boltzman Distributions

```
#--------- Boltzmann distribution -----------
#source: https://notebook.community/tommyogden/quantum-python-lectures/11_Monte-Carlo-Maxwell-Boltzmann-Distributions

import matplotlib.pyplot as plt
import numpy as np

def MB_speed(v,m,T):
    """ Maxwell-Boltzmann speed distribution for speeds """
    kB = 1.38e-23
    return (m/(2*np.pi*kB*T))**1.5 * 4*np.pi * v**2 * np.exp(-m*v**2/(2*kB*T))
```

```
fig = plt.figure()
ax = fig.add_subplot(111)

v = np.arange(0,800,1)
amu = 1.66e-27
mass = 85*amu

for T in [100,200,300]:
    fv = MB_speed(v,mass,T)
    ax.plot(v,fv,label='T='+str(T)+' K',lw=2)

ax.legend(loc=0)
ax.set_xlabel('$v$ (m/s)')
ax.set_ylabel('PDF, $f_v(v)$')
plt.savefig('Boltzmann.png', dpi=300)
plt.plot()
```

Python Codes for Poisson Distributions

- # Create separate plots for Poisson and Weibull distributions

- # Poisson distributions
- # Plotting Poisson distributions with three different lambda values

- # Define parameters
- lambdas_poisson = [1, 4, 10] # Different lambda values for Poisson distribution
- x_range_poisson = np.arange(0, 20) # X-range for plotting (discrete for Poisson)

- # Create plot for Poisson distributions
- fig, ax = plt.subplots(figsize=(7, 5))

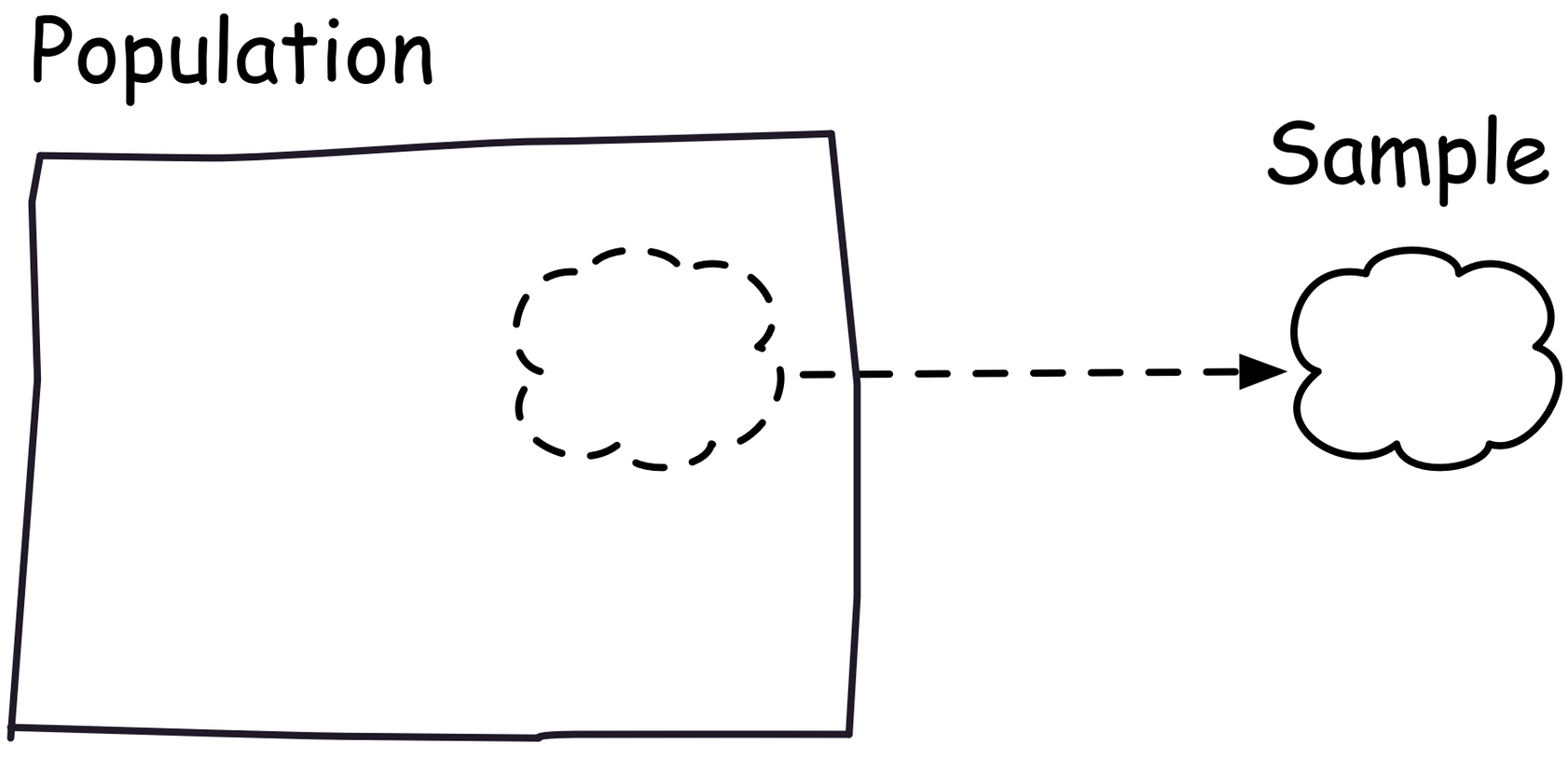
- for lam in lambdas_poisson:
-     poisson_pm = poisson.pmf(x_range_poisson, lam) # Poisson PMF
-     ax.plot(x_range_poisson, poisson_pm, '-o', label=f'λ = {lam}')

- ax.set_title('Poisson Distribution')
- ax.set_xlabel('Number of Events (x)')
- ax.set_ylabel('Probability')
- ax.legend()
- plt.figure(dpi=800)
- plt.show()

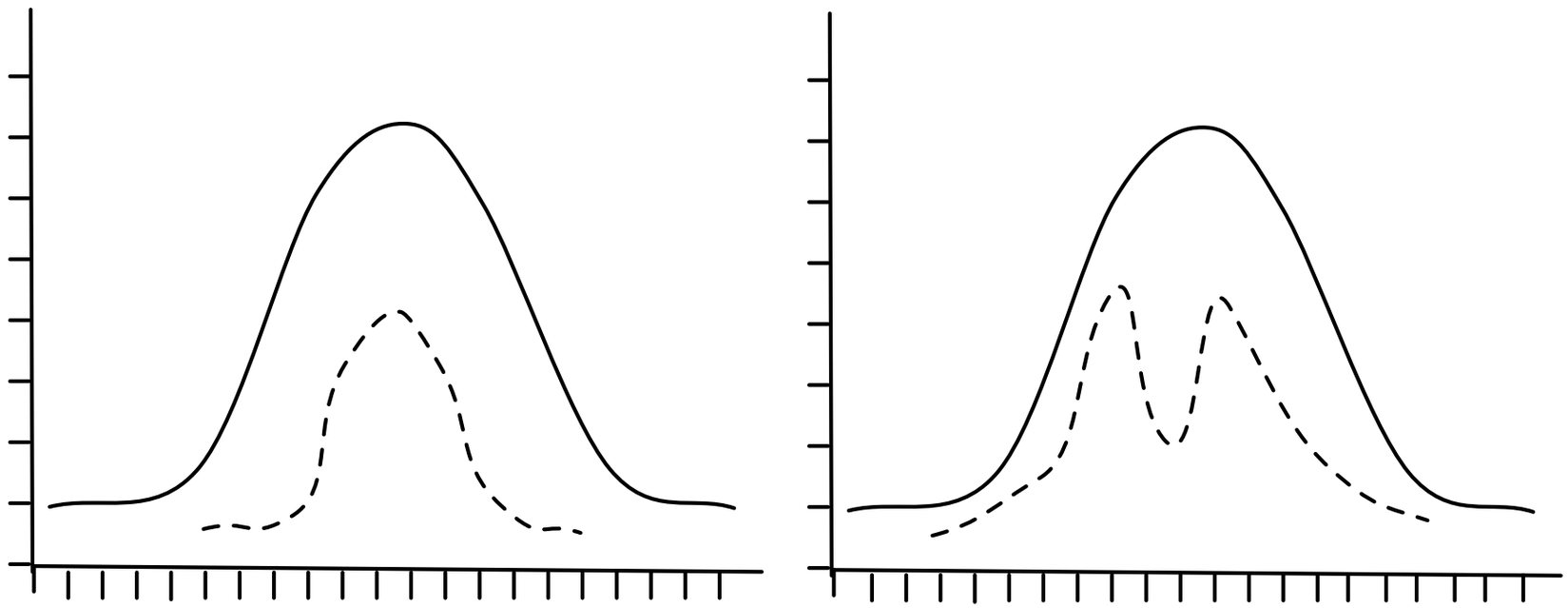
Python Codes for Weibull Distributions

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import weibull_min

k_values = [1, 2, 5]
lambda_values = [1, 1, 1]

x = np.linspace(0, 5, 100)

for i in range(len(k_values)):
    k = k_values[i]
    lambda_ = lambda_values[i]

    plt.plot(x, weibull_min.pdf(x, c=k, scale=lambda_),
             label='k={}, λ={}'.format(k, lambda_))

plt.xlabel('Number of Events (x)')
plt.ylabel('Probability')
plt.title('Weibull distribution')
plt.legend()
plt.show()
```

Real-World Setting



How much data is enough?



z

——— Original Population - - - - Sample

Given a sufficiently large sample size from a population with a finite level of variance, the mean of all samples from the same population will be approximately equal to the mean of the population.

Central Limit Theorem & The Law of Large Numbers

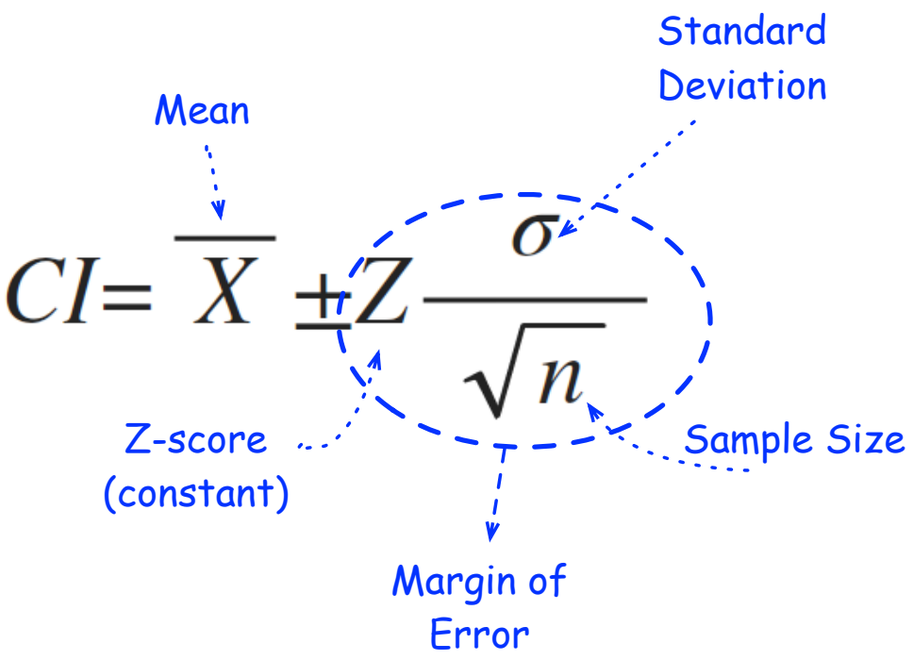
- *Central Limit Theorem* states if we collect enough amount of data the distribution tends toward a normal distribution even if the original variables themselves are not normally distributed.

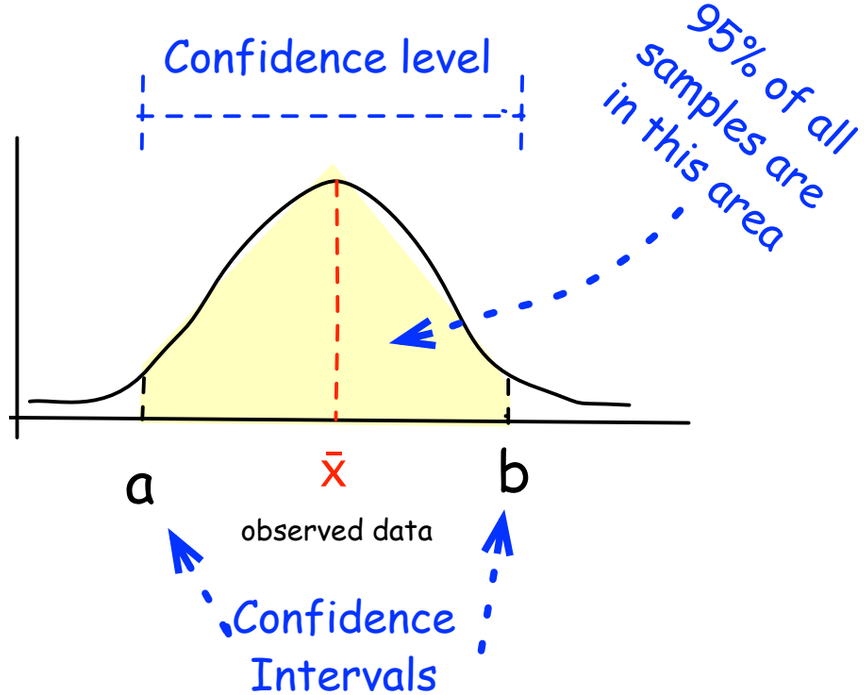
- The *Law of Large Numbers (LLN)* is a theorem that explains performing an experiment several times on the sample dataset brings the sample parameters (e.g. mean) closer to the population parameters, and as the number of experiments increases, the sample characteristics should get closer to the population.

How close is our sample data to the population?

Confidence Interval

- Confidence Interval is used to identify the **interval** or a **range of values** from sample to **estimate the chance whether our sample reflects the data in the population**. CI is operated based on a **confidence level**.
- The **confidence level** usually is 90%, 95% or 99%. There are other levels as well, but 95% is the most common used value for confidence level.





Normalization

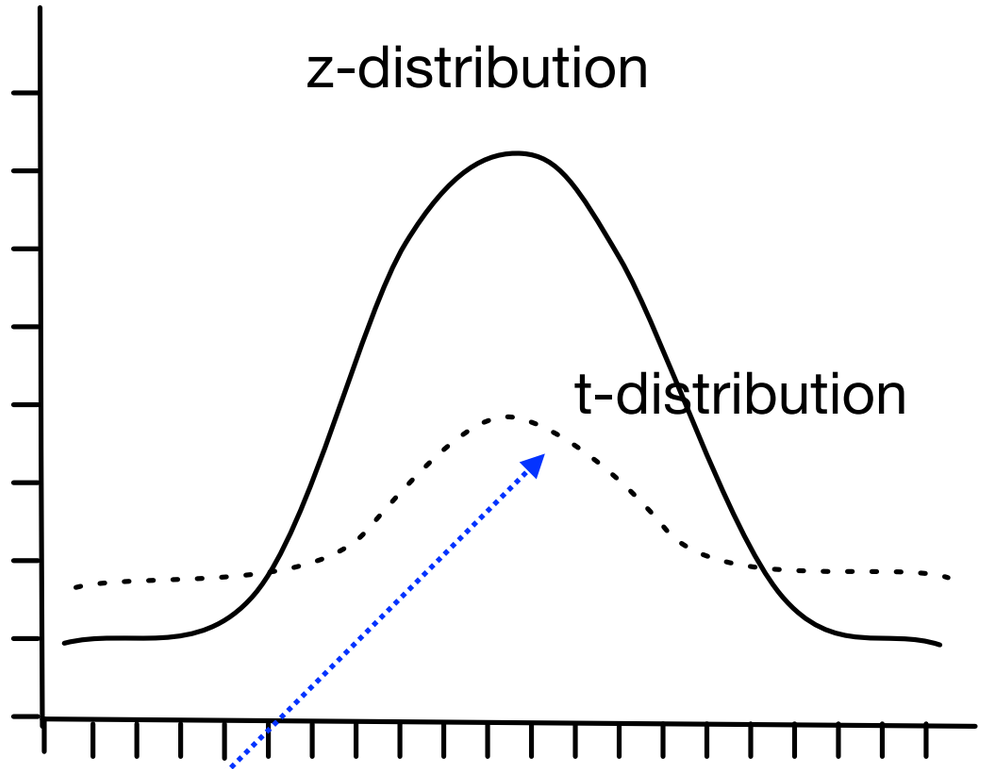
A single data point

Mean

z-score

$$z = \frac{x_i - \mu}{\delta}$$

Standard Deviation



mean = 0

SD = 1

Avocado Size Estimation

- 30 samples

- Average weight is 70gr SD= 20gr

95% CI [62.84, 77.16]

Increasing the Sample size

- 30 samples + 20 samples = 50 samples

- Average weight is 71gr (was 70gr) SD= 12gr (was 20gr)

95% CI [67.67, 74.33]

Optimal Sample Size

Based on a described equation for confidence interval we can **estimate a number of samples** we required. The equation to estimate *n* is as follows:

$$n \geq \left(\frac{Z . \sigma}{Margin \quad of \quad Error} \right)^2$$

Calculate a Confidence Interval for a Sample Size

- Suppose in the next two years, we sample 10 of you and ask you how much your salary annually increased because of learning this course. You give following numbers: {-1000, 15,000, 60,000, 30,000, 100,000, 40,000, 15,000, 35,000, 20,000, 36,000}

- Our goal is to find the confidence interval of 90% and 95% for the mean salary increase.

Calculate a Confidence Interval for a Given Sample Size

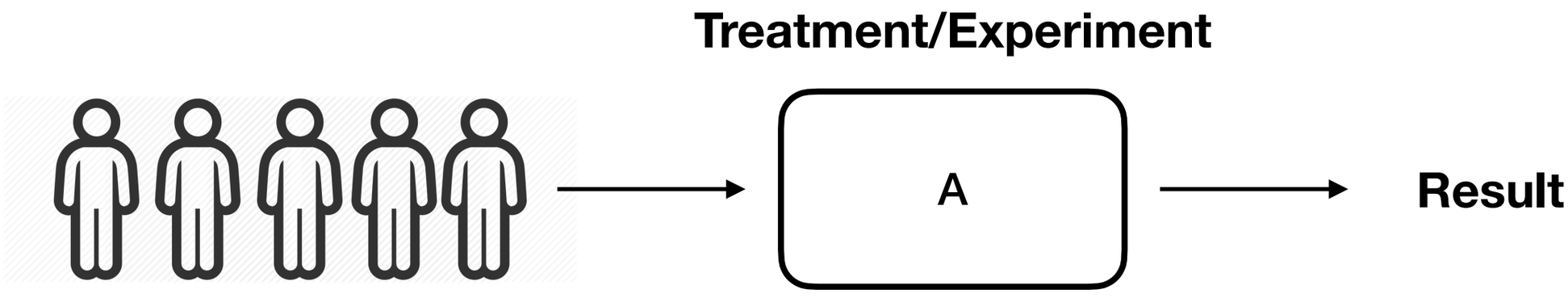
```
#------- Calculate Confidence Interval -------
import numpy as np
import scipy.stats as st

#define sample data
data = [-100, 150, 600, 300, 1000, 400, 150, 350, 200, 360]


#create 95% confidence interval for population mean weight
st.t.interval(alpha=0.95, df=len(data)-1, loc=np.mean(data),
scale=st.sem(data))
```

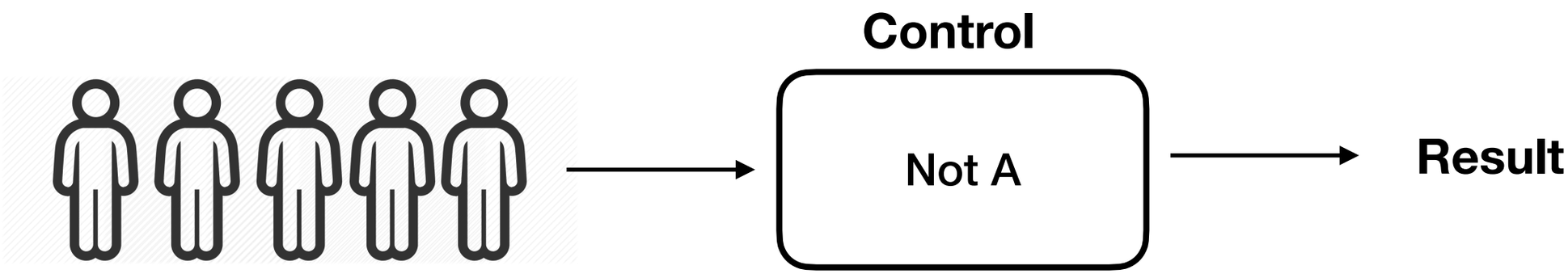
How to tell if two groups of data are significantly different?

How to tell if two groups of data are significantly different?

Significance Test

A/B Test





Hypothesis & *p*-Value

H_0 = Null hypothesis
H_1 = Alternate hypothesis

- H_1 is what we think should be correct, about the data, but H_0 says our hypothesis is wrong (H_1 is the hypothesis that says H_0 is false).

- Instead of proving H_1 in a statistical significance test we should reject H_0. In other words, to claim H_1 is true we must reject H_0.

Hypothesis & *p*-Value

H_0 = Null hypothesis
H_1 = Alternate hypothesis

- H_1 is what [illegible]bout the data, but H[illegible]ng (H_1 is the hypothe[illegible]

- Instead of [illegible]nificance test we sho[illegible] to claim H_1 is true we must reject H_0.

Our objective is to reject H_0

Test Outcome

“A significance test is comparing two groups of data together”

Test \ Reality	H0 = True	H0 = False
Accept H0	Correct Decision	Type II error
Reject H0	Type I error	Correct Decision

Hypothesis test outcomes.

Parametric vs None-Parametric

Parametric significance tests assume that all samples have a normal distribution.

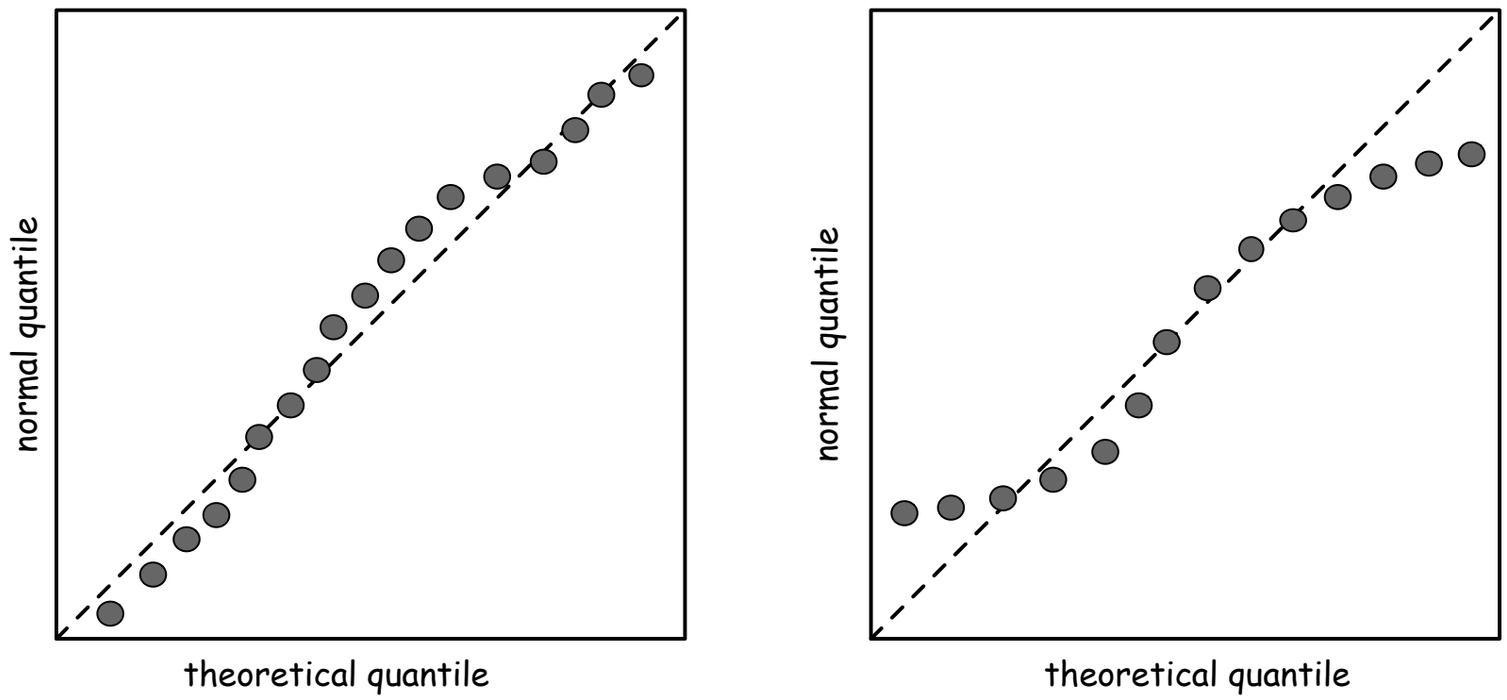
T-test, ANOVA, ANCOVA, MANOVA.

Non-parametric significance tests do not rely on the normal distribution of samples. Whenever, we encounter the term non-parametric, it is distribution free.

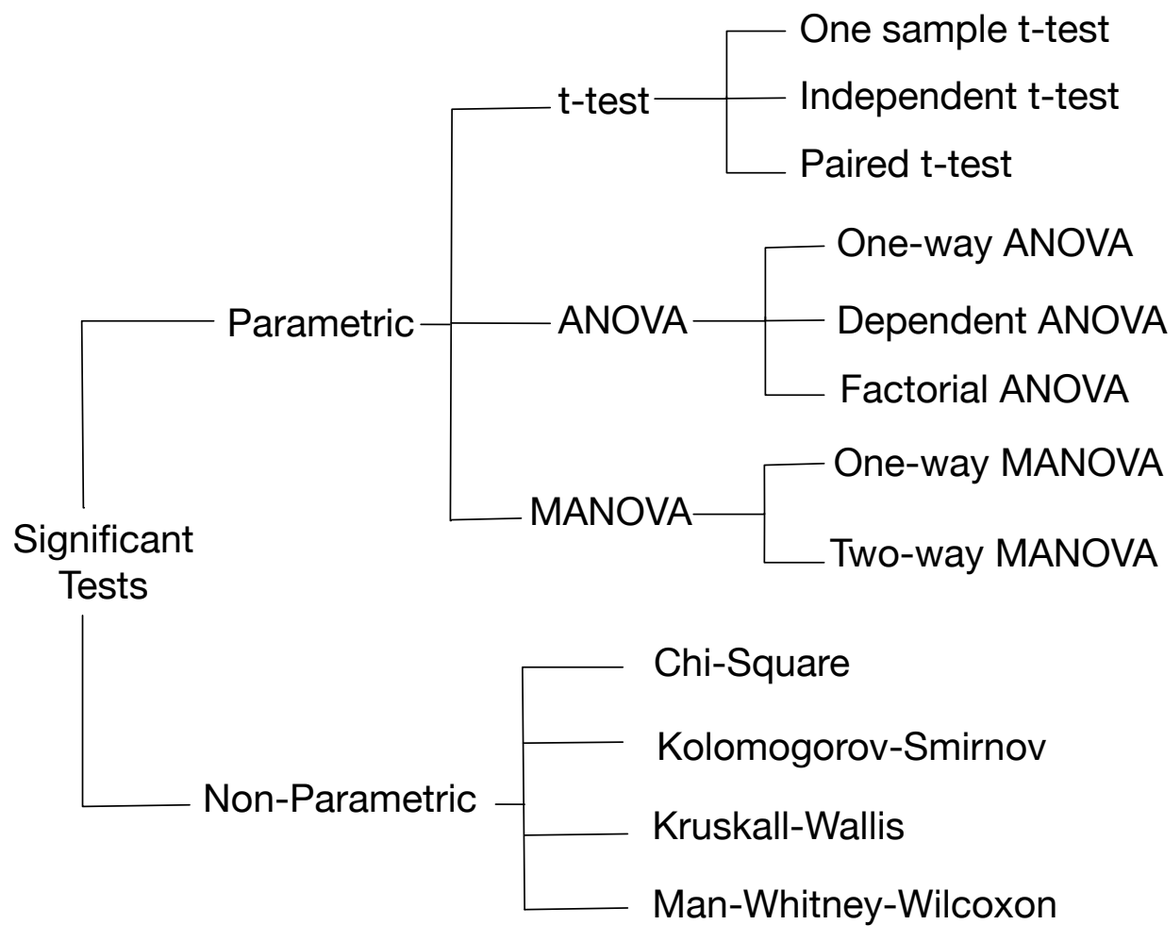
Chi-square test, Kolmogorov-Smirnov test, Kruskal-Wallis test, Mann-Whitney-U Test

How to check whether the distribution existed or not

Two Q-Q plots, the one on the left is close to a straight line, which means our dataset is following the desired distribution. The one on the right is not following the desired distribution because it is not creating a straight line.



Different Types of Significance Test

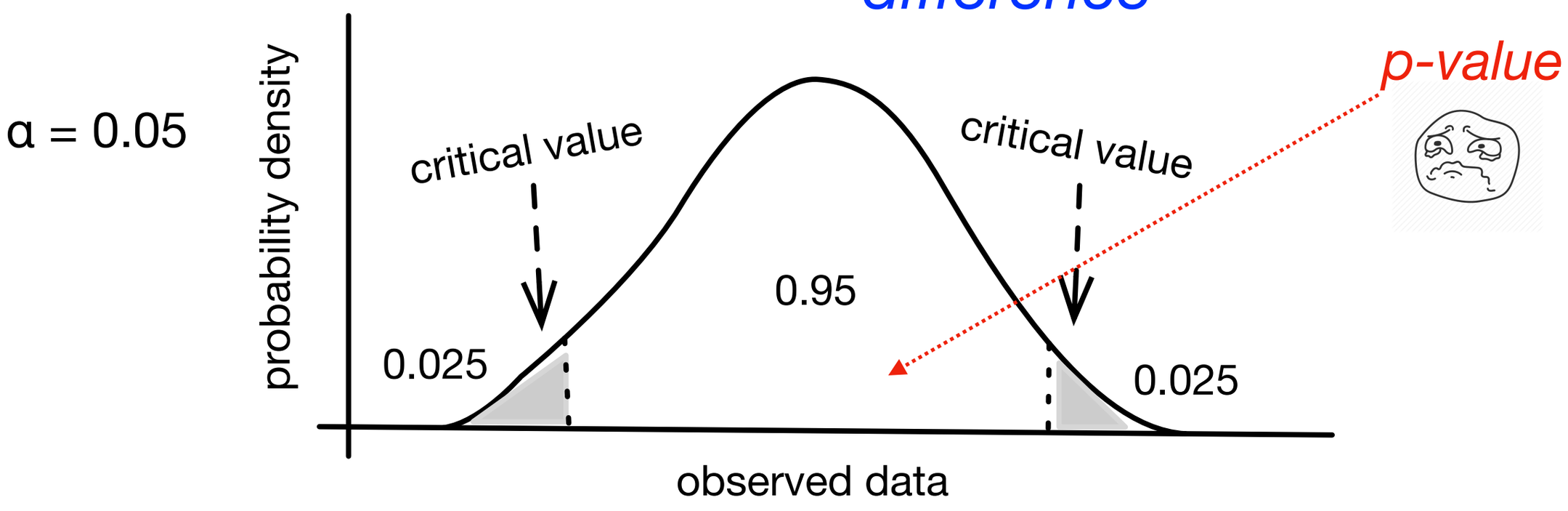


Significance Test

The purpose of significance test is to identify whether the differences between two groups of data we are comparing is by chance, or there is a significance difference.

if (p-value < α) —> there is a significant difference

if (p-value >= α) —> there is no significant difference



T-test

- **One-sample t-test:** it is used when we have **one group of sample data**, the sample size is small and we would like to compare it with a **known population mean**, but we do not know the **standard deviation** of a population.

- **Independent t-test:** We use this t-test to compare the mean of **two groups** which are **independent** and report whether there is a statistical significance among them. In addition to the normal distribution of data, the mean of both datasets should be different as well. Otherwise, H_0 will not be rejected, because H_0 assumes means of both dataset are equal.

- **Paired t-test:** It is used when we have **one group** of data, that is measured at **two different times**. It is another form of one-sample t-test. Usually, this test is being used to check if the new treatment, method, etc. is effective and works better than previous method or not.

ANOVA

- T-test is limited to two groups (sometimes one group in different conditions) comparison only. However, ANalysis Of VAriance (ANOVA) is a statistical method used to analyze differences among means of **two or more** groups of data.

- The H_0 in ANOVA assumes that all groups' mean is equal

- H_1 assumes that at least two of the group means are different.

H_0: $\mu_1 = \mu_2 = \mu_3$

H_1: Means are not all equal.

ANOVA

- ANOVA works with **factors (variables)** and **levels (values)**. Factors are “variables” such as gender and “levels” are possible values for these variables, i.e. male or female are values for the gender. The result of ANOVA test will be presented as a F-ratio. F-ratio is a ratio of two variances.

- **One-Way ANOVA**: In this test, we have **only one variable (factors)** with **at least two values (levels),** and levels are **independent**.

- **Repeated Measure (dependent) ANOVA**: In this model, we have only **one variable** with **at least two values,** but the values are **dependent**.

- **Factorial (two-way) ANOVA:** In factorial ANOVA, we have **more than two independent variables (factors),** and **levels (values) are dependent**. A well-known type of Factorial ANOVA is **Two-way ANOVA**.

Two-Way Anova is used when we have **one dependent value** and **two or more independent variables**.

ANOVA

- ANOVA works with **factors (variables)** and **levels (values)**. Factors are “variables” such as gender and “levels” are possible values for these variables, i.e. male or female are values for the gender. The result of ANOVA test will be presented as a F-ratio. F-ratio is a ratio of two variances.

- **One-Way ANOVA**: In this test, we have **only one variable (factors)** with **at least two values (levels),** and levels are **independent**.

- **Repeated Measure (dependent) ANOVA**: In this r **e variable** with **at least two values,** but the values are **dependent**.

Very similar to Paired T-Test

- **Factorial (two-way) ANOVA:** In factorial ANOVA, we have **more than two independent variables (factors),** and **levels (values) are dependent**. A well-known type of Factorial ANOVA is **Two-way ANOVA**.

Two-Way Anova is used when we have **one dependent value** and **two or more independent variables**.

ANOVA examples

- **One-Way ANOVA:** Measuring a blood pressure by using a drug XYZ. We give three different doses of drug XYZ to patients, i.e. 5 mcg, 10 mcg and 20 mcg. Analysis between these three groups could be done with One-Way ANOVA.
  variable or factor: *blood pressure,*
  value or level (independent): *Dosage in mcg.*

- **Repeated Measure** or **Dependent ANOVA:** Assume we measure the blood pressure of patients who have received 20 mcg of XYZ. If we measure their blood pressure in 3 different days, lets say Day 1, Day 3 and Day 6, in this case we need to use repeated measure ANOVA to see if there is any statistical difference between them.
  Patients are the same but their blood pressure is changing.
  factor: *blood pressure,*
  level (dependent): *Days.*

- **Factorial (two-way) ANOVA:** We are measuring the weight changes (dependent values) for males and females separately (independent variable) in two different age groups (more than 50 and less than 50), by using drug XYZ.
  For this types of analysis that we are dealing with two independent variables and different values (dependent), we use Factorial ANOVA.
  factor: *Age* ***AND*** *gender,*
  level (dependent): *Weight*

ANOVA examples

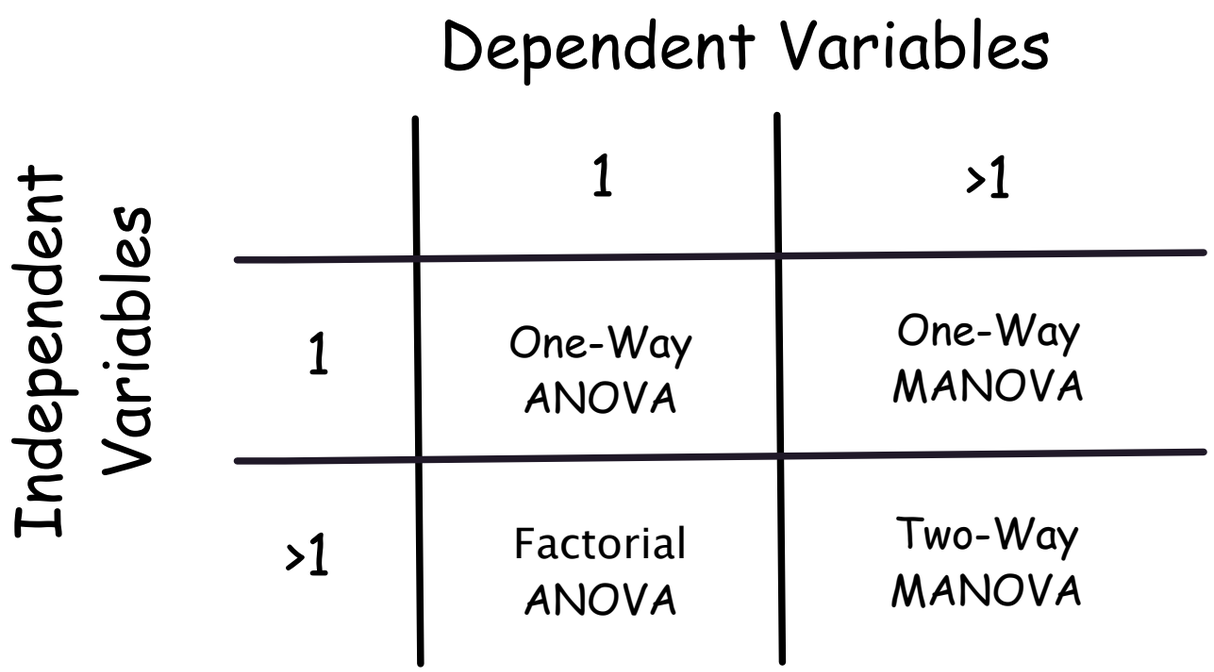
- For instance, assume we are measuring the **weight changes in different days** (dependent variable) for **male and female** separately (one independent variable).

- Or we are measuring mood changes (e.g. happy, sad,…), i.e. dependent variable, of male and female (one independent variable) to different dosage of testosterone (independent variable).

- For these types of analysis that we are dealing with two independent variables and different values (either dependent or independent), we use Factorial ANOVA.

MANOVA

- MANOVA (Multivariate ANalaysis Of VAriance) is a significance test for sample datasets that have **more than one dependent variables**, and **one independent variable**.
- **ANOVA is limited to one dependent variable** but MANOVA can handle more than one dependent variable.
- Similar to ANOVA, if there is one independent variable and more than one dependent variables we use One-Way MANOVA. If there is more than one independent variables and more than one dependent variables we use Two-Way MANOVA.



ANCOVA

- ANCOVA (ANalysis of COVAriance). There are variables that are not independent nor dependent variable, but they have effect on the dependent variable, these variables are called **covariate** or **nuisance**. Covariate is a type of control variable. For instance, weather can have an affect on mood, but we did not consider it while measuring the blood pressure and mood.

- The goal of a scientific process is to establish a relation between independent variable and dependent variable without any external influence.

- Just remember: When we have covariate instead of ANOVA we should use ANCOVA. The same is applicable for MANOVA and MANCOVA, when we need MANOVA but we have covariates we go for MANCOVA.

Non-Parametric Tests

χ^2 Test

- Chi-Square (χ^2) Test
  - Test of Independence between two categorical variable.
  - Test for the Goodness-of-fit

- This test operates based on a contingency table. **Contingency**, **Crosstab** or **RxC table** (read as R by C table, R stayed for row and C for column) is a table that presents frequency of different variables in a dataset and their relations together.

- It calculates the *p*-value based on the differences between observed and expected values. It is quit easy to calculate expected value as following:

$$E = \frac{total\ row \times total\ column}{sample\ size}$$

Chi-Square: Test of independence

- It calculates the *p*-value based on the differences between observed and expected values. It is quit easy to calculate expected value as following:

$$E = \frac{total\ row \times total\ column}{sample\ size}$$

- After we have calculated the expected values, we can use the following formula to calculate the chi-square score:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Observation

Avocado Weight	Taste: Good	Taste: Too Oily	Taste: Too Crunchy	**Total**
=0.2	47	14	22	**83**
>0.2	8	34	12	**54**
<0.2	3	1	20	**24**
Total	**58**	**49**	**52**	**161**

Expectation

Avocado Weight	Taste: Good	Taste: Too Oily	Taste: Too Crunchy	**Total**
=0.2	(83x58)/161 = 29.9	(83x49)/161 = 25.6	(83x52)/161 = 26.8	**83**
>0.2	(54x58)/161 = 19.4	(54x49)/161 = 16.4	(54x52)/161 = 17.44	**54**
<0.2	(24x58)/161 = 8.6	(24x49)/161 = 7.3	(24x52)/161 = 7.7	**24**
Total	**58**	**49**	**52**	**161**

Chi-Square: Goodness of Fit

- Goodness-of-Fit is a test used to check how well a **sample (observed)** dataset fits an **expected (hypothesized) distribution**.

$$Goodness - of - Fit : \chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

- Based on what we have explained we can say the Goodness-of-fit test has following hypothesis:

H_0*:* The sampled (observed) dataset is **not** significantly different than the expected dataset.

H_1: The sampled (observed) dataset is significantly different than the expected dataset.

Chi-Square: Goodness of Fit

- Goodness-of-Fit is a test used to check how well a **sample (observed)** dataset fits an **e** **ion**.

It is a special case of significance test, we compare observation with the expectation. But the expectation is extracted from the observation. This test refers to goodness-of-fit.

$$Goo \qquad \frac{E_i)^2}{}$$

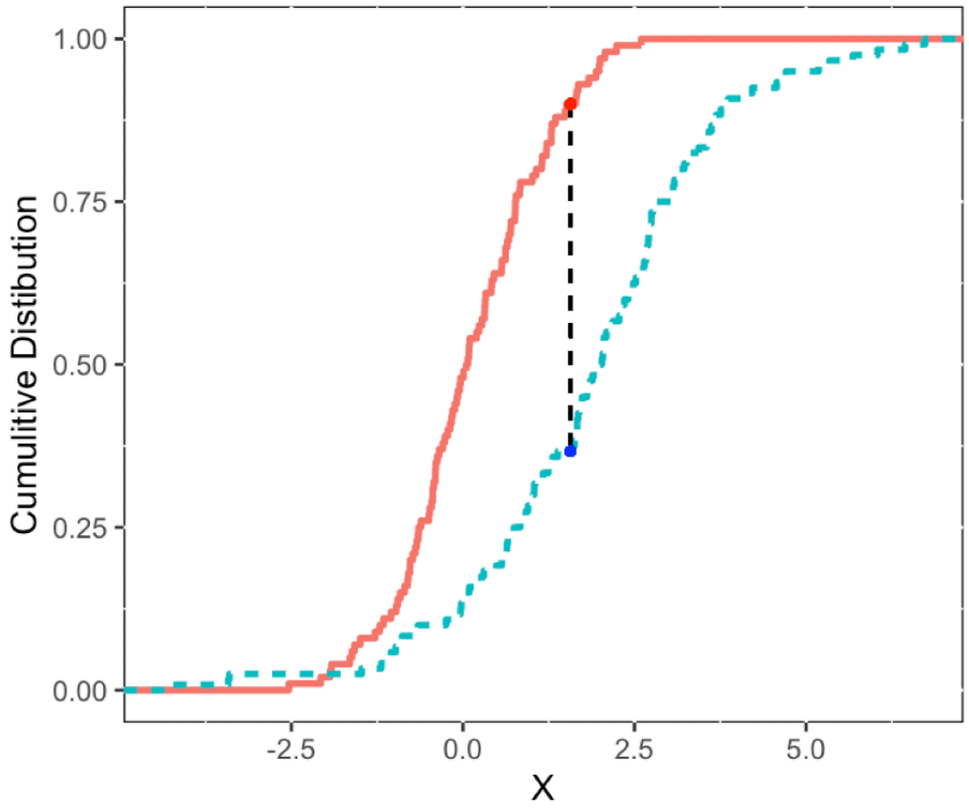
- Based on what we have explained we can say the Goodness-of-fit test has following hypothesis:

H_0*:* The sampled (observed) dataset is **not** significantly different than the expected dataset.

H_1: The sampled (observed) dataset is significantly different than the expected dataset.

Kolmogorov-Smirnov (KS-Test)

It is based on the measuring the differences between Cumulative Distribution Function (CDF) of two different dataset.



KS-Test based on comparison of the distance between CDFs of two samples.

Sample Significant Test

```
#------------ Significance test -------
import numpy as np
from scipy.stats import ttest_ind
from scipy.stats import kstest

v1 = np.random.normal(size=100)
v2 = np.random.normal(size=100)

# t-test
res = ttest_ind(v1, v2)
print(res)

#---------------------------------

v3 = np.random.chisquare(df=10, size=100)
# ks-test
res = res = kstest(v1, v2)
print(res)
```

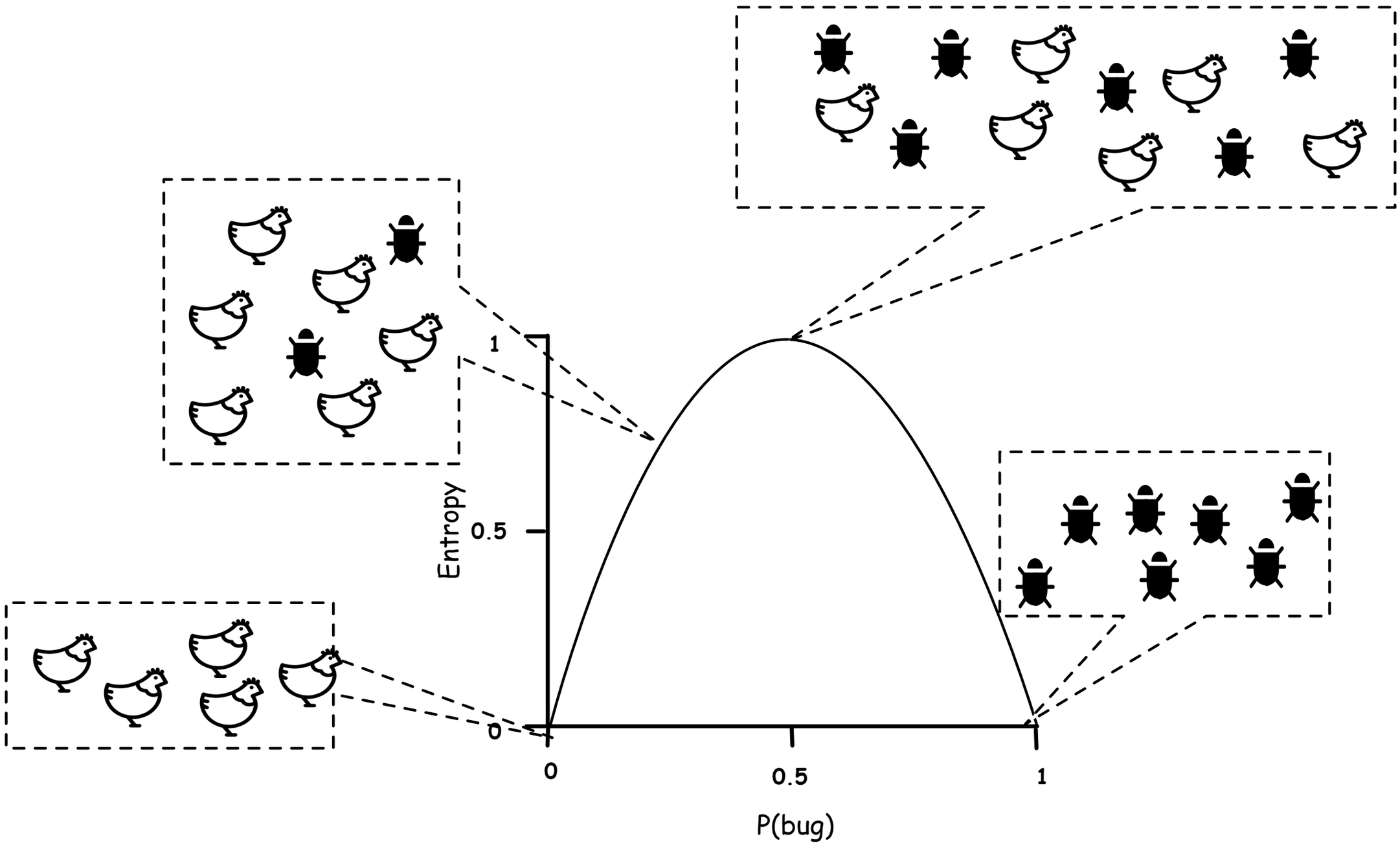
Entropy and Information Gain

Entropy

- Entropy is a measurement of disorder or measurement of impurity. In other words, entropy is a measure of the uncertainty of a variable [Shanon ’49].

- Some literature [Bishop ‘06] defines entropy as *the average amount of information needed to specify the state of a random variable*.

Visualization of Entropy



Entropy

- The entropy of random variable *X*, with *n* outcomes can be defined as follows:

number of outcomes or events

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 P(x_i)$$

Entropy of Variable X

Probability of Variable having the value of x_i

Entropy

- The entropy of random variable X with n outcomes can

Erwin Schrödinger, an Austrian physicist, who is one of the founders of quantum mechanics, states:

a hallmark of a living creature is to reduce its entropy by increasing entropy around itself.

In other words, he stated *the total entropy should increase, but it allows decreases in some places as long as it is increasing elsewhere*.

Entropy

- The higher the entropy, the more information we need to be able to make a valid justification.

- We can refer to entropy as a number, which explains *how unpredictable is our probability distribution.*

Entropy Calculation

- Assume the bug inside chicken room is a sign of impurity,

- We sample some data from each room and we have the followings probabilities:

Room 1: P(bug)=0.5, p(chicken)=0.5 —> Entropy = - [0.5 $\log_2(0.5)$ + 0.5 $\log_2(0.5)$] = 1
Room 2: P(bug)=0.65, p(chicken)=0.35 —> Entropy = - [0.65 $\log_2(0.65)$ + 0.35 $\log_2(0.35)$] = 0.93
Room 3: P(bug)=0.75, p(chicken)=0.25 —> Entropy = - [0.75 $\log_2(0.75)$ + 0.25 $\log_2(0.25)$] = 0.81
Room 4: P(bug)=0.95, p(chicken)=0.05 —> Entropy = - [0.95 $\log_2(0.95)$ + 0.05 $\log_2(0.05)$] = 0.29

This means that room 1 is the most impure one because its entropy is equal to one, which is maximum. Room 4 has the purest boxes because its entropy is the lowest, 0.29.

Entropy Calculation

```
#------- Calculating Entropy of different trials -----

from scipy.stats import entropy

print (entropy([1/2, 1/2], base=2)) # this is the
highest possible entropy between two trial

print (entropy([1/6, 1/6, 1/6, 1/6, 1/6, 1/6],
base=6))

print (entropy([1/6, 3/6, 1/6, 1/6], base=6))

print (entropy([1/6, 4/6, 1/6], base=6))
```

Cross Entropy AND Kullback-Leibler-Divergence (KL-Divergence, relative Cross Entropy)

Cross entropy measures the difference between two probability distributions over the same set of events. Given two probability distributions, *p* and *q,* over a set of *I events*, the cross entropy between them is defined as $H(p, q)$ and computed as:

$$H(p, q) = -\sum_i p_i \log_2(q_i)$$

$$H(p, q) = D_{KL}(p \,||\, q) + Entropy(p)$$

The KL-Divergence between two discrete distributions *p* and *q* is calculated as as follows:

$$D_{KL}(p \,||\, q) = H(p, q) - H(p) = \sum_i p_i \, log(\frac{p_i}{q_i})$$

The KL-Divergence between two continuous distributions *p* and *q* is calculated as follows:

$$D_{KL}(p \,||\, q) = \int_i p_i \, log(\frac{p_i}{q_i})$$

Jensen Shanon Divergence (JS-Divergence)

- KL-Divergence can be used to measure distances between two distributions, but it is not symmetric and does not have triangle inequality i.e.,
  $D_{KL}(p||q) \neq D_{KL}(q||p)$.

- Besides, KL-Divergence range is unbounded. It means that it varies between 0 to ∞ (when two distributions do not have any overlap).

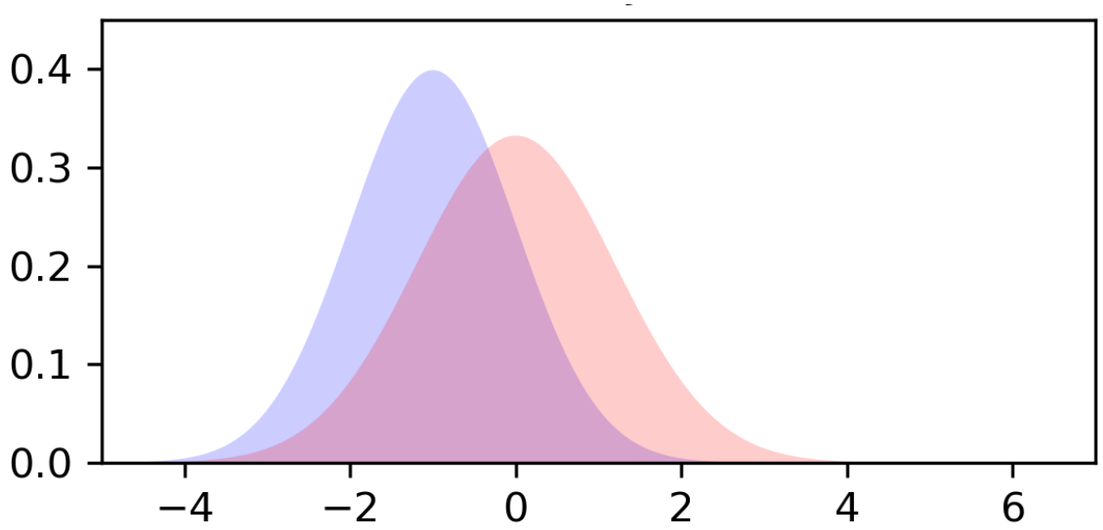
Jensen Shanon Divergence (JS-Divergence)

- To handle these limitations, we can use *Jensen Shanon Divergence (JS-Divergence)*. It is symmetric, bounded, and written based on KL-Divergence as follows:

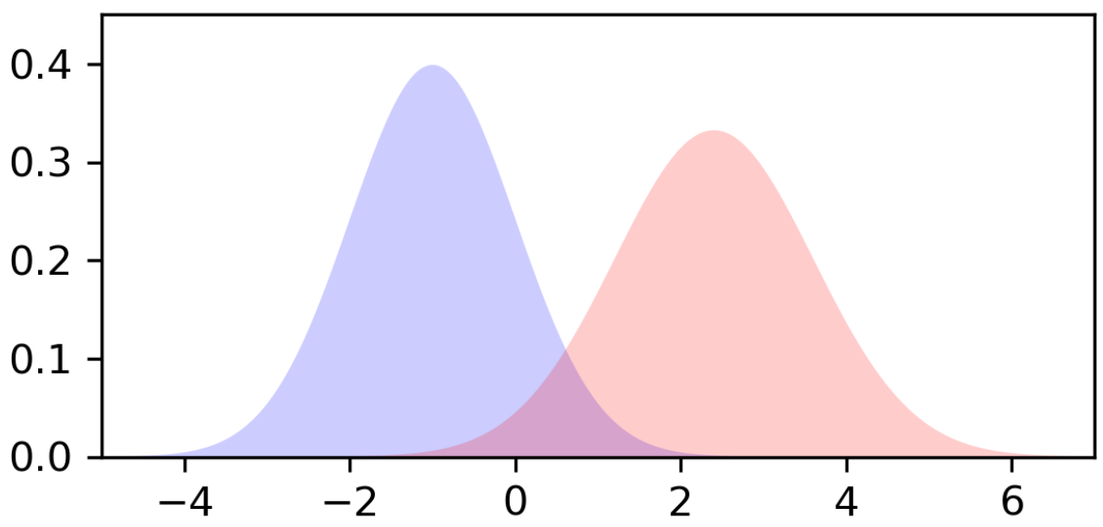
$$JS(p,q) = \frac{1}{2}KL(p||m) + \frac{1}{2}KL(q||m)$$

- where *m* is the average of *p* and *q*, $m_i = (p_i + q_i)/2$.

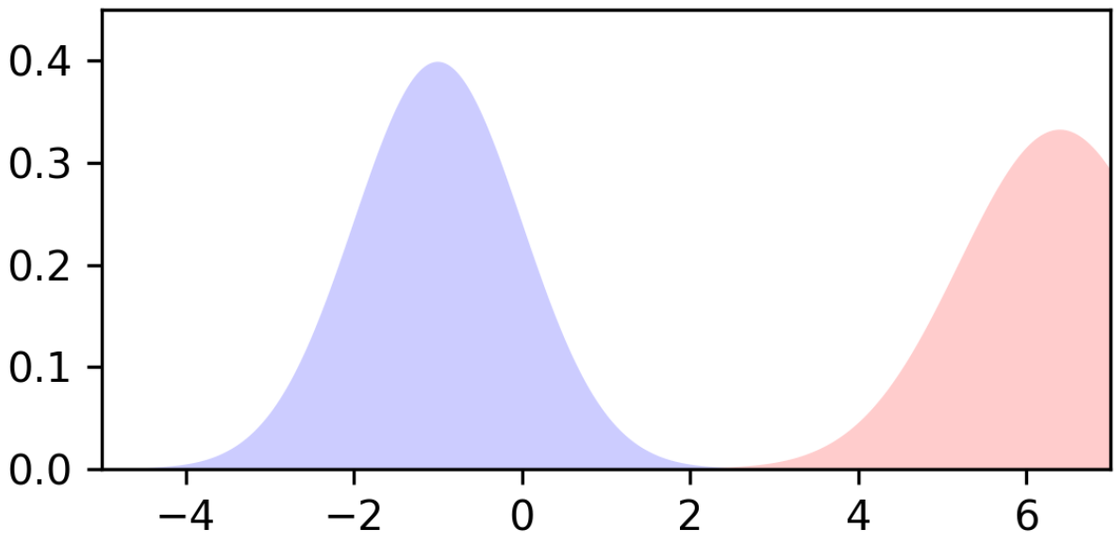
- The range of JS is between 0 ($p = q$) to 1 (*p* and *q* do not have any overlap at all).



KL-Divergence (p,q) = 0.38
KL-Divergence (q,p) = 0.38
JS-Divergence (p,q) = 0.09
JS-Divergence (q,p) = 0.09



KL-Divergence (p,q) = 4.04
KL-Divergence (q,p) = 5.83
JS-Divergence (p,q) = 0.54
JS-Divergence (q,p) = 0.54



KL-Divergence (p,q) = ∞
KL-Divergence (q,p) = ∞
JS-Divergence (p,q) = 0.70
JS-Divergence (q,p) = 0.70

KL-Divergence and JS-Divergence

```
import numpy as np
from scipy.stats import norm
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()

#------- Simple equation to calculate KL-div -------
def kl_divergence(pk, qk):
  # arraynise
  pk = np.asarray(pk)
  # normalise
  pk = 1.0*pk / np.sum(pk, axis=0)
  qk = 1.0*qk / np.sum(qk, axis=0)
  # check to decide if we apply single or multi entropy
  if qk is None:
      return np.sum(entropy_single(pk), axis=0)
  else:
      # arraynise
      qk = np.asarray(qk)
      if len(qk) != len(pk):
          raise ValueError("qk and pk must have same length.")
      qk = 1.0*qk / np.sum(qk, axis=0)
      return np.sum(entropy_multi(pk, qk), axis=0)

#------- Simple equation to calculate JS-div -------
def js_divergence(p,q):
    m = (p+q)/2  # mean of them
    js = (kl_divergence(p, m) + kl_divergence(q, m))/2
    return js
```

KL-Divergence and JS-Divergence

```
plt.figure(figsize=(18,5))

x = np.arange(-10, 10, 0.0001)
p = norm.pdf(x, -4, 1)
q = norm.pdf(x, 4, 1.1)
plt.fill_between(x, p, facecolor="b", alpha=0.2)
plt.fill_between(x, q, facecolor="r", alpha=0.2)

print("KL(P||Q):" , str(kl_divergence(p, q)) )
print("KL(Q||P):" , str(kl_divergence(q, p)) )
print("JS(Q||P) or JS(P||Q):", js_divergence(p,q))
```

KL-Divergence and JS-Divergence

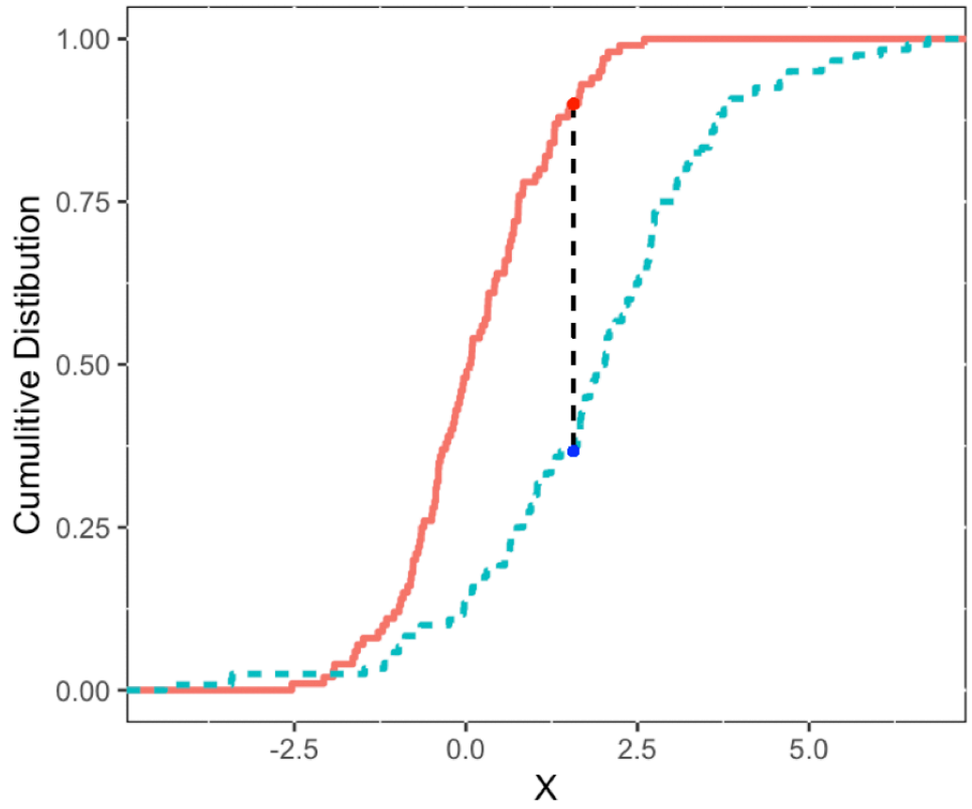
```
x = np.arange(-10, 10, 0.0001)
p = norm.pdf(x, 3, 1)
q = norm.pdf(x, 3, 1.3)
```

```
plt.fill_between(x, p, facecolor="b", alpha=0.2)
plt.fill_between(x, q, facecolor="r", alpha=0.2)
```

```
print("KL(P||Q):" , str(kl_divergence(p, q)) )
print("KL(Q||P):" , str(kl_divergence(q, p)) )
print("JS(Q||P) or JS(P||Q):", js_divergence(p,q))
```

Kolmogorov-Smirnov

It is based on the measuring the differences between Cumulative Distribution Function (CDF) of two different dataset.



KS-Test based on comparison of the distance between CDFs of two samples.

Summary of How to Compare Distributions

- Kolmogorov-Smirnov (Also used as non-parametric significant test)
- Cross Entropy
- Kullback-Lieber divergence (KL-Divergence)
- Jensen Shanon Divergence (JS-Divergence)

Probability Estimation

Probability vs Likelihood

- **Probability** means what is the chance of observing *X* in the given sample dataset. This means that in a given dataset, what is the chance of observing *X*?

- **Likelihood** means given the observed subset *X* of a dataset, what is the best distribution parameters, that fit the given *X*.

In simple words, Probability is a measure of *how likely a particular event will occur*.

The likelihood, on the other hand, is a measure of *how well a particular set of data points fits a specific model or hypothesis*.

Maximum Likelihood Estimation (MLE) Approach

- There is a baseline method to estimate the probability of data, Maximum Likelihood Estimation, and an approach to implement it is Expectation Maximization.

- For example, we have a small part of a dataset and assume the original dataset has a distribution (e.g., Gaussian distribution). We don't know the parameters (mean and standard deviation) of that Gaussian distribution because there could be infinite numbers of Gaussian distributions.

- The MLE is a procedure that tries to identify the mean μ and standard deviation, σ, by using the sample dataset and constructs a distribution that is the closest match to the original dataset distribution.

- The *goal* of MLE is to find the best distribution parameters that fit the original dataset (population).

The MLE function is shown as a function ℓ. To find the population dataset distribution, there is an unknown set of parameters θ that the distribution of the dataset depends on these parameters. The function that identifies the likelihood is called the likelihood function and it is shown as L_n. We use a function called the maximum likelihood function to determine θ, and it is presented as $\ell(\theta)$. Formally the MLE for the observed dataset *X* can be written as:

$$\hat{\ell}(\theta; X) = \arg\max_{\theta} \hat{L}_n(\theta; X)$$

";" is a sign of conditional probability and is often stated using the semicolon ";" instead of the "|" because θ is an unknown parameter. Nevertheless, if you see some literature use "|" instead, that is also correct. When you encounter the ";" sign, it means it is something that the output of the algorithm will be specified (e.g. predict it).

Usually, for the sake of computational efficiency, instead of maximum likelihood, we use the logarithm of likelihood (log-likelihood). The logarithm is a concave function, which means that the log-likelihood function has a unique maximum, making the optimization problem well-behaved.

Therefore, log-likelihood can be written as:

$$\hat{\ell}(\theta; X) = ln[L_n(\theta; X)] \text{ or } \hat{\theta} = \arg\max_{\theta} ln[L_n(\theta; X)]$$

Even we can use negative log-likelihood:

$$\hat{\ell}(\theta; X) = -ln[L_n(\theta; X)] \text{ or } \hat{\theta} = \arg\min_{\theta} -ln[L_n(\theta; X)]$$

It means we calculate the inverse of minimization, which is equal to maximization. In simple words, minimizing the error is equivalent to maximizing the log likelihood.

Expectation Maximization

- **Latent variable problem:** Although MLE does not have access to the original dataset, it assumes the dataset is complete or fully observed. Nevertheless, part of the dataset could be missing in the observation subset that is used by MLE. Those missing parts could construct parameters that are known as latent variables (latent variables refer to parameters that do not exist in the observed dataset).

- The expectation Maximization (EM) algorithm is an algorithm that implements the MLE, when there is a latent variable in the dataset.

- The EM algorithm is recommended to use for MLE implementation when we are dealing with missing data in our observed dataset.

- In summary, EM is an algorithm that implements the MLE approach.

EM Algorithm

- The objective of EM, similar to the MLE objective, is to find unknown parameters θ that find the best distribution fit to the original dataset (approximate maximum likelihood).

- **E-step** makes an initial guess of parameters for the expected distribution.

- **M-step** starts after the E-step, and when newly observed data are fed into the model. In this step, the EM algorithm tweaks the estimated parameters (from E-Step) to cover newly observed data.

- From M-step, the process will be repeated until the created distribution does not change in E-step or M-step and it reaches a stable state (converged), or a maximum threshold of iteration reaches.

EM Algorithm Example

```
# source: https://www.kaggle.com/code/charel/learn-by-example-expectation-maximization/notebook
# ------ libraries
# plotting
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("white")
%matplotlib inline
#for matrix math
import numpy as np
#for normalization + probability density function computation
from scipy import stats
#for data preprocessing
import pandas as pd
from math import sqrt, log, exp, pi
from random import uniform
```

EM Algorithm Example

```
# generate data by giving two means and SD to have two Normal Dist.
random_seed=36788765
np.random.seed(random_seed)

Mean1 = 2.0  # Input parameter, mean of first normal probability distribution
Standard_dev1 = 4.0 #@param {type:"number"}
Mean2 = 9.0 # Input parameter, mean of second normal  probability distribution
Standard_dev2 = 2.0 #@param {type:"number"}
```

```
# generate data
y1 = np.random.normal(Mean1, Standard_dev1, 1000)
y2 = np.random.normal(Mean2, Standard_dev2, 500)
data=np.append(y1,y2)

# For data visiualisation calculate left and right of the graph
Min_graph = min(data)
Max_graph = max(data)
x = np.linspace(Min_graph, Max_graph, 2000) # to plot the data

print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("1", Mean1, Standard_dev1))
print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("2", Mean2, Standard_dev2))
sns.histplot(data, bins=20, kde=False);
```

EM Algorithm Example

```
class Gaussian:
    "Model univariate Gaussian"
    def __init__(self, mu, sigma):
        #mean and standard deviation
        self.mu = mu
        self.sigma = sigma

    #probability density function
    def pdf(self, datum):
        "Probability of a data point given the current parameters"
        u = (datum - self.mu) / abs(self.sigma)
        y = (1 / (sqrt(2 * pi) * abs(self.sigma))) * exp(-u * u / 2)
        return y

    def __repr__(self):
        return 'Gaussian({0:4.6}, {1:4.6})'.format(self.mu,self.sigma)
print("Gaussian class defined")
```

EM Algorithm Example

```
class GaussianMixture_self:
    "Model mixture of two univariate Gaussians and their EM estimation"

    def __init__(self, data, mu_min=min(data), mu_max=max(data), sigma_min=1, sigma_max=1, mix=.5):
        self.data = data
        #todo the Algorithm would be numerical enhanced by normalizing the data first, next do all the EM steps and do the de-
normalising at the end

        #init with multiple gaussians
        self.one = Gaussian(uniform(mu_min, mu_max),
                            uniform(sigma_min, sigma_max))
        self.two = Gaussian(uniform(mu_min, mu_max),
                            uniform(sigma_min, sigma_max))

    #as well as how much to mix them
    self.mix = mix

def Estep(self):
    "Perform an E(stimation)-step, assign each point to gaussian 1 or 2 with a probability"
    # compute weights
    self.loglike = 0. # = log(p = 1)
    for datum in self.data:
        # unnormalized weights
        wp1 = self.one.pdf(datum) * self.mix
        wp2 = self.two.pdf(datum) * (1. - self.mix)
        # compute denominator
        den = wp1 + wp2
        # normalize
        wp1 /= den
        wp2 /= den     # wp1+wp2= 1, it either belongs to gaussian 1 or gaussion 2
        # add into loglike
        self.loglike += log(den) #freshening up self.loglike in the process
        # yield weight tuple
        yield (wp1, wp2)
```

EM Algorithm Example

```
def Mstep(self, weights):
    "Perform an M(aximization)-step"
    # compute denominators
    (left, rigt) = zip(*weights)
    one_den = sum(left)
    two_den = sum(rigt)

    # compute new means
    self.one.mu = sum(w * d  for (w, d) in zip(left, data)) / one_den
    self.two.mu = sum(w * d  for (w, d) in zip(rigt, data)) / two_den

    # compute new sigmas
    self.one.sigma = sqrt(sum(w * ((d - self.one.mu) ** 2)
                              for (w, d) in zip(left, data)) / one_den)
    self.two.sigma = sqrt(sum(w * ((d - self.two.mu) ** 2)
                              for (w, d) in zip(rigt, data)) / two_den)
    # compute new mix
    self.mix = one_den / len(data)


def iterate(self, N=1, verbose=False):
    "Perform N iterations, then compute log-likelihood"
    for i in range(1, N+1):
        self.Mstep(self.Estep()) #The heart of the algorith, perform E-stepand next M-step
        if verbose:
            print('{0:2} {1}'.format(i, self))
    self.Estep() # to freshen up self.loglike

def pdf(self, x):
    return (self.mix)*self.one.pdf(x) + (1-self.mix)*self.two.pdf(x)

def __repr__(self):
    return 'GaussianMixture({0}, {1}, mix={2.03})'.format(self.one,
                                                          self.two,
                                                          self.mix)
```

EM Algorithm Example

```
# See the algorithm in action
n_iterations = 20
best_mix = None
best_loglike = float('-inf')
mix = GaussianMixture_self(data)
for _ in range(n_iterations):
    try:
        #train!
        mix.iterate(verbose=True)
        if mix.loglike > best_loglike:
            best_loglike = mix.loglike
            best_mix = mix

    except (ZeroDivisionError, ValueError, RuntimeWarning): #
Catch division errors from bad starts, and just throw them out...
        print("one less")
        pass
```

EM Algorithm Example

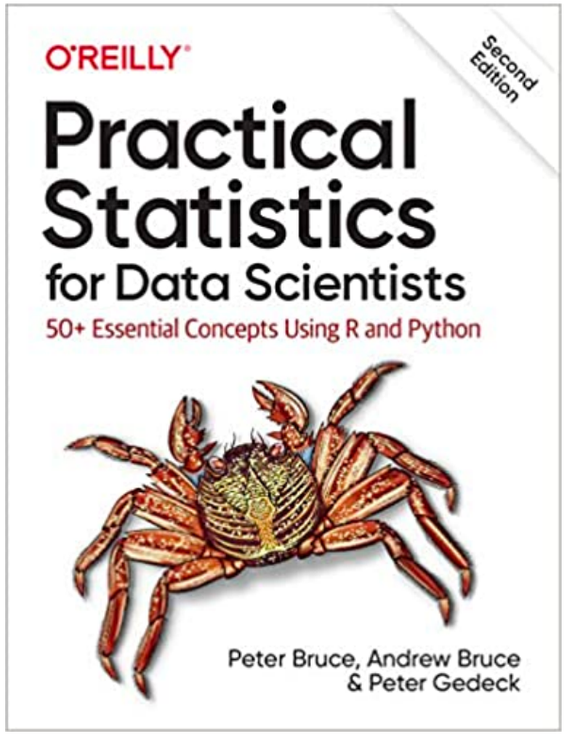
```
# Find best Mixture Gaussian model
n_iterations = 50
n_random_restarts = 4
best_mix = None
best_loglike = float('-inf')
print('Computing best model with random restarts...\n')
for _ in range(n_random_restarts):
    mix = GaussianMixture_self(data)
    for _ in range(n_iterations):
        try:
            mix.iterate()
            if mix.loglike > best_loglike:
                best_loglike = mix.loglike
                best_mix = mix
        except (ZeroDivisionError, ValueError, RuntimeWarning): # Catch division errors from bad starts, and just throw
them out...
            pass
#print('Best Gaussian Mixture : μ = {:.2}, σ = {:.2} with μ = {:.2}, σ = {:.2}'.format(best_mix.one.mu, best_mix.one.sigma,
best_mix.two.mu, best_mix.two.sigma))

print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("1", Mean1, Standard_dev1))
print('Input Gaussian {:}: μ = {:.2}, σ = {:.2}'.format("2", Mean2, Standard_dev2))
print('Gaussian {:}: μ = {:.2}, σ = {:.2}, weight = {:.2}'.format("1", best_mix.one.mu, best_mix.one.sigma, best_mix.mix))
print('Gaussian {:}: μ = {:.2}, σ = {:.2}, weight = {:.2}'.format("2", best_mix.two.mu, best_mix.two.sigma, (1-
best_mix.mix)))
#Show mixture
sns.distplot(data, bins=20, kde=False, norm_hist=True);
g_both = [best_mix.pdf(e) for e in x]
plt.plot(x, g_both, label='gaussian mixture');
g_left = [best_mix.one.pdf(e) * best_mix.mix for e in x]
plt.plot(x, g_left, label='gaussian one');
g_right = [best_mix.two.pdf(e) * (1-best_mix.mix) for e in x]
plt.plot(x, g_right, label='gaussian two');
plt.legend();
```

References

* Penn State Stat-500 and MiniTab:
https://onlinecourses.science.psu.edu/stat500 Another good source that I would say it is a free online book for statistic. They also provide a minitab (statistical software) examples for their code. The minitab weblog, http://blog.minitab.com/, has fantastic descriptions as well and we have used it a lot for writing this section. If you are not can not afford to pay the license fee, there are wide availability of free R and Python statistical packages and you can use them.

* Stats How To:
http://www.statisticshowto.com
This a web page with short and clear description of statistical information. We have used this page as well to check the validity of our example. The good thing with this page is that you can learn your required content in a short amount of time.

* Vassar Stats:
http://vassarstats.net
This is another useful webpage that you can add your numbers and do the calculation online for you. Of course these pages are mostly for teaching purposes and when you have a large amount of data you should use a software packages, such as R, SPSS, Python or minitab.



Source codes

https://github.com/Rezar/MLBook

Codes for Python

https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/