Probability Distributions

- gives the possibility of each outcome of a random experiment or event
- measure of uncertainty of various phenomena
- statistical function that describes the likelihood of obtaining all possible values that a random variable can take

1. Discrete Probability Distributions
2. Continuous Probability Distributions

Discrete Probability Distributions

- used for discrete data

1. Binomial Distribution
2. Poisson Distribution
3. Hypergeometric Distribution

Binomial Distribution

- for `only two possibilites`, to model binary data, such as coin tosses
- `probability of occurence` should be `known already`
- binomial distribution is `sampled with replacement`
- population size is `finite`
- used when you have a `fixed number of independent trials` (experiments) with `two possible outcomes` (usually referred to as success and failure), and you want to know the `probability of getting a certain number of successes` in those trials

Q. probability of rains in June on any given day is 0.4

- I am in Mumbai for 20 days. What are the chances that

1. it rains for exactly 12 days
2. it rains for max 12 days
3. it rains for min 12 days

scipy.stats.binom

```
import scipy
from scipy import stats
from scipy.stats import binom
```

binom.pmf(exact number, max possible number, probability)

```
# for exactly 12 days
binom.pmf(12, 20, 0.4)
# PMF: Probability Mass Function
# pmf(exact number, max possible number, probability)
# for exactly 12 days out of 20 days of stay with probability 0.4
```

```
0.03549743955648174
```

binom.cdf(range number, max possible number, probability)

```
# for at max 12 days
binom.cdf(12, 20, 0.4)
# CDF: Cummulative Distribution Function
# cdf(range number, max possible number, probability)
# for 0days or 1 days or 2 days or ... or 12 days out of 20 days stay with probability 0.4
```

```
0.9789710725222288
```

```
binom.cdf(4, 20, 0.4)
```

```
0.05095195319416651
```

```
binom.pmf(0, 20, 0.4)+binom.pmf(1, 20, 0.4)+binom.pmf(2, 20, 0.4)+binom.pmf(3, 20, 0.4)+binom.pmf(4, 20, 0.4)
# same as binom.cdf(4, 20, 0.4)
# prob(0 days) + prob(1 days) + prob(2 days) + ... + prob(4 days)
```

```
0.05095195319416648
```

```
# for 13days, 14days, ..., 20 days
1 - binom.cdf(12, 20, 0.4)
```

```
0.021028927477771187
```

```
1 - binom.cdf(11, 20, 0.4)
# for min 12 days (12 days, 13 days, 14 days, ... 20 days)
```

```
0.05652636703425307
```

binom.sf(range number - 1, max possible number, probability)

```
binom.sf(11, 20, 0.4)
# SF: survival Function
# SF = 1 - CDF
```

```
0.056526367034253025
```

Poisson's Distribution

- when `probability is too low` or `cannot be calculated`
- we work with `average number of occurences`, `not probability`, to model count data
- describes `probabilities for counts of events` that occur in a specified observation space
- `population size` has no limits, it can be `infinite`
- e.g. probability of lightening striking at a place

- A variable follows a Poisson distribution when the following conditions are true:
  - Data are `counts of events`.
  - All `events are independent`.
  - The `average rate of occurrence does not change` during the period of interest.

Note

- as probability tends to zero, binomial distribution tends to be poisson's distribution

Q. Number of accidents in an area are 30 per day

1. exactly 27 accidents take place
2. max 27 accidents take place
3. min 27 accidents take place

scipy.stats.poisson

```
import scipy
from scipy import stats
from scipy.stats import poisson
```

poisson.pmf(exact number, average count)

```
poisson.pmf(27, 30)
# PMF: Probability Mass Function
# pmf(exact number, average count)
# exactly 27 accidents
```

```
0.06553248388325897
```

poisson.cdf(range number, average count)

```
poisson.cdf(27, 30)
# CDF: Cummulative Distribution Function
# cdf(range number, average count)
# max 27 acidents with average 30 accidents per day
```

```
0.3328690840455234
```

```
1 - poisson.cdf(27, 30)
# min 28 accidents with average 30 accidents per day
# 28 or 29 or 30 or 31 .... n accidents
```

```
0.6671309159544766
```

```
1 - poisson.cdf(26, 30)
# min 27 accidents with average 30 accidents per day
# 27 or 28 or 29 or 30 or 31 .... n accidents
```

```
0.7326633998377348
```

Hypergeometric Distribution

- is a discrete probability distribution that calculates the likelihood an event happens `k` times in `n` trials when you are `sampling from a small population without replacement`

- it is `like the binomial distribution` except for the `sampling without replacement` aspect
- `hypergeom.cdf(k, N, K, n)`
    - `K` : outcome of interest in Population
    - `N` : Population Size
    - `k` : outcome of interest in Sample
    - `n` : Sample Size
- used when sampling is done without replacement, but binomial distribution is sampled with replacement
- `population size` is `finite`
- used in industrial quality checking

Q. 70 LEDs from the manufacturer , out of which upto 7 can be defective

- we take a sample of 20, out of which UPTO 2 can be allowed to be defective and we still accept the lot

scipy.stats.hypergeom

```
import scipy
from scipy import stats
from scipy.stats import hypergeom
```

hypergeom.cdf(k, N, K, n)

```
hypergeom.cdf(2, 70, 7, 20)
# cdf(no of events, population size, population of events, sample size)
# cdf(no of allowed events in sample, population size, no of allowed events in population, picked sample size)
# prob(0 defective) + prob(1 defective) + prob(2 defective) + prob(3 defective)
```

```
0.6842509992202706
```

hypergeom.pmf(k, N, K, n)

```
hypergeom.pmf(0, 70, 7, 20)+hypergeom.pmf(1, 70, 7, 20)+hypergeom.pmf(2, 70, 7, 20)
# same as hypergeom.cdf(2, 70, 7, 20)
```

```
0.6842509992202704
```

Continuous Probability Distributions

- used for continuous data

1. Exponential Probability Distributions
2. Normal Probability Distributions
3. Student's t Probability Distributions

Exponential Probability Distributions

- used when we wish `to model the time gap between two successive events`
- usually `failures are taken as events` in industry, and use this distribution to `model failure times`
- `small values are more likely` than larger values
- `MTBF` should be `known already` and the `time gap to find the failures` should be `known already`
    - `MTBF` = `Mean` / `Average Time Between Failures`

Q. On an average, there are 200 server trips in one year. A client is visiting the office for 60 hours.

- What are the chances that he sees the server trip ?

```
365*24
# 365d X 24hrs = 8460hrs
```

```
8760
```

```
8760/200
# 8760hrs / 200 trips per year
# next trip after 43.8hrs
```

```
43.8
```

```
e = 2.7182
# base of natural log
```

Find e^(-a/b)

- `e^(-a/b)`
- `e^(-timeFrame/avgFailureTime)`

```
e**(-60/43.8)
# e^(-a/b)
# e^(-timeFrame/avgFailureTime)
# probability that the client will not see the failure
```

```
0.2541522515269206
```

1 - e^(-a/b)

- Exponential Probability Distribution, that failure will be observed

```
1 - e**(-60/43.8)
# prob that the client will see the breakdown
```

```
0.7458477484730794
```

Normal Probability Distributions

- also called `Gaussian Distribution`, `symmetric around mean`, indicating data near the mean is frequent than data far from the mean
- normal distribution will be `used only when` `sample size > 30`, or `for historical/huge data`
- A `Normal bell-shaped curve` is `defined by two parameters`: `Mean` & `Standard Deviation`
    - For `Standard Normal Curve`, `Mean = 0`, `S.D. = 1`
- `Normal Distribution used when sample size > 30`, and if `Sample Size <= 30, then use Student's t-distribution` [as per Central Limit Theorem]
    - if `Population S.D.` is `known already`, then `use Normal Distribution`
    - if `Population S.D.` is `not known already`, then `use Student's t-distribution`

Q. averge time travel = 70min with SD of 2min

what is the probability of reaching CST

1. in less than 67min

2. more than 67min

    - `X` outcome of interest = 67
    - `x̄` Sample Mean = 70
    - `σ` S.D. = 2

scipy.stats.norm

```
import scipy
from scipy import stats
```

```
from scipy.stats import norm
```

norm.cdf(X, x̄, σ)

```
norm.cdf(67, 70, 2)
# norm.cdf(X, x̄, σ)
# probability of less than 67min
```

```
0.06680720126885807
```

```
1 - norm.cdf(67, 70, 2)
# probability of more than 67min
```

```
0.9331927987311419
```

norm.sf(X, x̄, σ)

```
norm.sf(67, 70, 2)
# norm.sf(X, x̄, σ)
# probability of more than 67min
# same as 1 - norm.cdf(67, 70, 2)
```

```
0.9331927987311419
```

norm.cdf(Z)

- uses Z-Statistics to calculate normal distribution

```
norm.cdf(-1.5)
# norm.cdf(Z)
# calculating normal distribution using Z-Statistics
```

```
0.06680720126885807
```

Z-Score or Z-Statistics , Z

- used to `determine` whether to `reject the null hypothesis or otherwise`
- value on `x-axis` of `Standard Normal Distribution`
- directly `linked with C.L.`
- if `C.L. increases`, `Z-Score also increases`
- `Z` = `(X - Mean)/S.D.`
- `Z` = `(X - μ)/σ`
- `Z` = `(Observation - x̄) / S.D.`
- `Z` = `(X - x̄) / σ`

```
x̄ (Sample Mean) = 70
X (Claimed / Population Mean)= 67
σ (S.D.) = 2
```

```
Z = (X - x̄) / σ
Z = (67 - 70) / 2
Z = -3 / 2
Z = -1.5
```

Student's t Probability Distributions

- symmetrical, bell-shaped distribution, similar to the standard normal curve, but has heavier tail
- shape of the t-distribution varies with the change in degrees of freedom

- mean & S.D. are needed to transform Z-Statistics
- also needs degree of freedom(dof), higher the degrees of freedom (dof), the closer this distribution will approximate a standard normal distribution with a mean of 0 and a S.D. of 1
- Normal Distribution used when sample size > 30, and if Sample Size <= 30, then use Student's t distribution [as per Central Limit Theorem]
- Population S.D. is known, then use Normal Distribution and if Population S.D. is not known, then use Student's t distribution

Applications of student's t-distribution

- The important applications of t-distributions are as follows:
  - Testing for the hypothesis of the population mean
  - Testing for the hypothesis of the difference between two means. In this case, the t-test can be calculated in two different ways, such as
    - Variances are equal
    - Variances are unequal
  - Testing for the hypothesis of the difference between two means having the dependent sample
  - Testing for the hypothesis about the Coefficient of Correlation. It is involved in three cases. They are:
    - When the population coefficient of correlation is zero, i.e. $\rho = 0$.
    - When the population coefficient of correlation is not zero, i.e. $\rho \neq 0$.
    - When the hypothesis is examined for the difference between two independent correlation coefficients

Q. averge time travel = 70min with SD of 2min

what is the probability of reaching CST

1. in less than 67min
2. more than 67min

above data is based on 20 trips(< 30), so we need to use student's t distribution

scipy.stats.t

```
import scipy
from scipy import stats
from scipy.stats import t
```

t.cdf(Z, n-1)

- uses Z-statistics `Z` & Degree of fredom `dof` to calculate Student's t distribution
- does not take S.D. or σ in arguments

```
t.cdf(-1.5, 20-1)
# t.cdf(Z, dof)
# uses Z-statistics to calculate Student's t distribution
```

```
0.07502426537113577
```

```
# n = 200
# Z = -1.5
t.cdf(-1.5, 200-1)
# t.cdf(Z, dof)
```

```
0.06759961872077877
```

```
t.ppf(0.06759961872077877, 200-1)
```

```
-1.499999999999639
```

Central Limit Theorem (CLT)

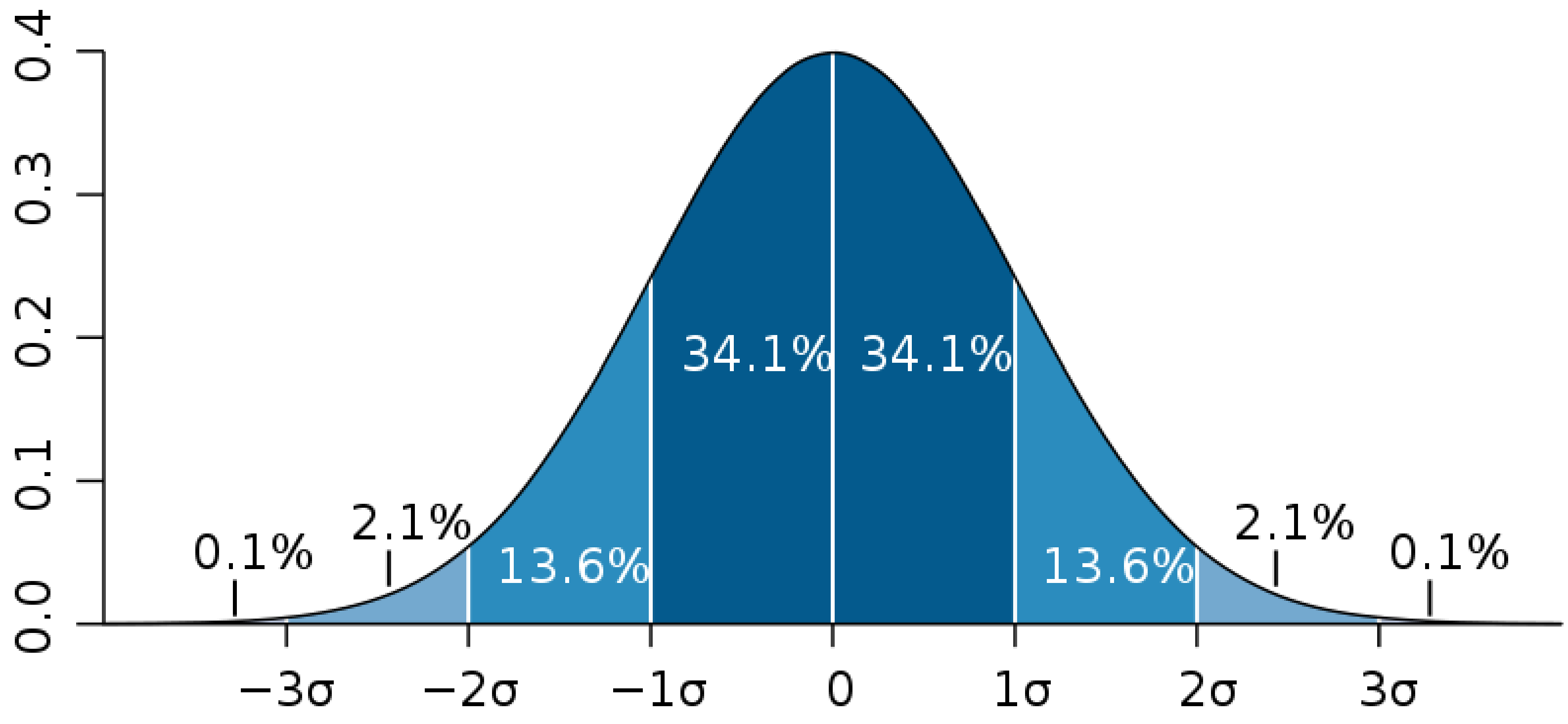
- `Given a dataset wih unknown distribution (it could be uniform, binomial, or completely random) the sample means will approximate the normal distribution`
- applicable to almost all types of probability distributions(with finite variances), except cauchy's distribution because it has infinite varianes
- states that given a sufficently large sample size, the distribution of the sample mean for a variable will approximate a normal distribution regardless of that variable in the population distribution
- the sampling distribution of the mean will always be normally distributed, as long as the sample size is large enough
- as the number of samples/ sample size increase, the distribution of sample means will get closer to the normal distribution
- Sample mean = Population mean =
- μ μ Sample standard deviation = (Population standard deviation) / √n
- μ μ Sample standard deviation = σ / √n

Conditions of Central Limit Theorem

- it states that sampling distribution of mean will always follow a normal distribution under following conditions
  1. `sample size` is `sufficiently large`, usually sample size `n >= 30`
  2. `samples` are `independent` and `identically distributed` (i.e. random variables), means `sampling` should `be random`
  3. `population` distribution has `finite variance`, it is not applicable to distribution with infinite variance

Empirical Rule / Three-Sigma Rule

1. `1-σ` : 68.26% of the observations will fall between +/- 1 S.D. from the mean
2. `2-σ` : 95.44% of the observations will fall between +/- 2 S.D. from the mean
3. `3-σ` : 99.73% of the observations will fall between +/- 3 S.D. from the mean



scipy.stats.norm

```
import scipy
from scipy import stats
from scipy.stats import norm
```

norm.cdf(Z)

```
norm.cdf(-1)
# norm.cdf(Z)
```

```
# uses Z-Statistics from normal distribution
# gives z_statistics on negative-side of mean only
```

```
0.15865525393145707
```

```
norm.cdf(-1)*2
# multiplying Z-Statistics on negative-side with 2 to get
# value from both sides of mean
```

```
0.31731050786291415
```

1-σ

```
1 - norm.cdf(-1)*2
# 1-σ
# use Z-Statistics of -1 to find value on both sides of mean,
# and subtract from total probabitliy of 1 to get 1-σ
```

```
0.6826894921370859
```

2-σ

```
1 - norm.cdf(-2)*2
# 2-σ
# use Z-Statistics of -2 to find value on both sides of mean,
# and subtract from total probabitliy of 1 to get 2-σ
```

```
0.9544997361036416
```

3-σ

```
1 - norm.cdf(-3)*2
# 3-σ
# use Z-Statistics of -3 to find value on both sides of mean,
# and subtract from total probabitliy of 1 to get 3-σ
```

```
0.9973002039367398
```

→ Inferential Statistics

- analyze samplings to make predictions about larger populations
- allows you to make inferences
- by taking a small sample instead of working on the whole population
- explains the chance of occurrence of an event
- achieved by probability

5 elements of inferential statistics

1. population size
2. number of variables
3. sample set
4. satistical inference about the population
5. measure of reliability

Sample Statistics, Population Parameter & Chance Variation

1. Population: entire set
2. Sample: subset of population

1. Population Parameter (PP): properties related to pupulation set

2. Sample Statistics (SS): properties related to sample set
3. meaning of bias: prejudice / assumption
4. chance variation: changes due to sampling, can be removed to zero only if we take population
5. `Sample Statistic(SS) = Population Parameter(PP) + Bias + Chance Variation`

```
Sample Statistic(SS) = Population Parameter(PP) + Bias + Chance Variation
seen or shown        = reality                  + Bias + Chance variation
```

- Note: Sample Statistics changes from sample to sample, but Population Parameter remains same for the same dataset

Confidence Level (C.L.)

- probability of delivering correct result
- default value of C.L. is 95%
- `C.L. of Sample < 100%`
- `C.L. of Population =100%`
- `C.L. ∝ Sample Size (n)`
- to `increase confidence level`, `increase Sample Size`

```
import scipy
from scipy import stats
```

scipy.stats.t.interval(C.L., dof, x̄, StdErrMean)

- calculating the interval, its size, and the factors it depends on
- `bigger the interval size`, `lesser the C.L.`

for C.L. = 0.95 or 95 %

- Sample Size `n` = 2000
- SS or Mean(x) or `x̄` = 27,000
- S.D. `σ` = 5000
- Standard Error of the Mean, `StdErrMean = S.D. / √SampleSize`
- Standard Error of the Mean, `StdErrMean = σ / √n`

```
x = stats.t.interval(0.95, 2000-1, 27000, 5000/2000**0.5)
# interval(C.L., dof, x̄, StdErrMean)
# Standard Error of the Mean = S.D. / sqrt(SampleSize)
# returns (lower_value, higher_value)
x
```

```
(26780.73660551642, 27219.26339448358)
```

```
base_interval = x[1] - x[0]
base_interval
# Interval Size
```

```
438.5267889671595
```

for C.L. = 0.99 or 99.9 %

- Sample Size `n` = 2000
- SS or Mean(x) or `x̄` = 27,000
- S.D. `σ` = 5000
- `Standard Error of the Mean, StdErrMean = S.D. / √SampleSize`
- `Standard Error of the Mean, StdErrMean = σ / √n`

```
x = stats.t.interval(0.999, 2000-1, 27000, 5000/2000**0.5)
# interval(C.L., dof, x̄, StdErrMean)
x
```

```
(26631.56301484492, 27368.436985155084)
```

```
CL_interval = x[1]- x[0]
CL_interval
```

```
736.8739703101637
```

for C.L. = 0.95 or 95 %

- Sample Size `n` = 2000
- SS or Mean(x) or `x̄` = 27,000
- S.D. `σ` = 1000 [5000-->1000]
- `Standard Error of the Mean, StdErrMean = S.D. / √SampleSize`
- `Standard Error of the Mean, StdErrMean = σ / √n`

```
x = stats.t.interval(0.95, 2000-1, 27000, 1000/2000**0.5)
# interval(C.L., dof, x̄, StdErrMean)
x
```

```
(26956.147321103283, 27043.852678896717)
```

```
Sigma_interval = x[1] - x[0]
Sigma_interval
```

```
87.70535779343481
```

for C.L. = 0.95 or 95 %

- Sample Size `n` = 2000000 [2000->2000000]

- SS or Mean(x) or `x̄` = 27,000
- S.D. `σ` = 5000
- `Standard Error of the Mean, StdErrMean = S.D. / √SampleSize`
- `Standard Error of the Mean, StdErrMean = σ / √n`

```
x = stats.t.interval(0.95, 2000000-1, 27000, 5000/2000000**0.5)
# interval(C.L., dof, x̄, StdErrMean)
x
```

```
(26993.070476684625, 27006.929523315375)
```

```
ss_interval = x[1] - x[0]
ss_interval
```

```
13.859046630750527
```

```
#                               Confidence Level        σ           n       IntervalSize
# Base                          95                      5000        2000    438.5
# CL 95-->99.9                  99.9                    5000        2000    736.8
# Sigma 5000-->2000             95                      1000        2000    087.7
# SS or Mean 2000-->2000000     95                      5000        2000000 13.85
```

Deciding Minimum Sample Size, n

- `n = 9(z ** 2) x σ ** 2) / d ** 2`
- `n` : Minimum Sample Size
- `Z` : Z-Statistics or Z-Score, value on x-axis of Standard Normal Deviation [directly linked with C.L., if C.L. increases, Z also increases] [provided by Boss in your organization]
- `σ` : S.D. [prior known]
- `d` : Interval Size (`MoE` : Margin of Error) [by client of your organization]

Proportion

- S.D. `σ = sqrt(p(1 - p))`