Inferential Statistics

Population:

A complete collection of the objects or measurements is called the population or else everything in the group we want to learn about will be termed as population. or else In statistics population is the entire set of items from which data is drawn in the statistical study. It can be a group of individuals or a set of items.

Population is the entire group you want to draw conclusions about.

The population is usually denoted with N

1. The number of citizens living in the State of Rajasthan represents a population of the state
2. All the chess players who have FIDE rating represents the population of the chess fraternity of the world
3. the number of planets in the entire universe represents the planet population of the entire universe
4. The types of candies and chocolates are made in India.

population mean is usually denoted by the Greek letter μ

Sample:

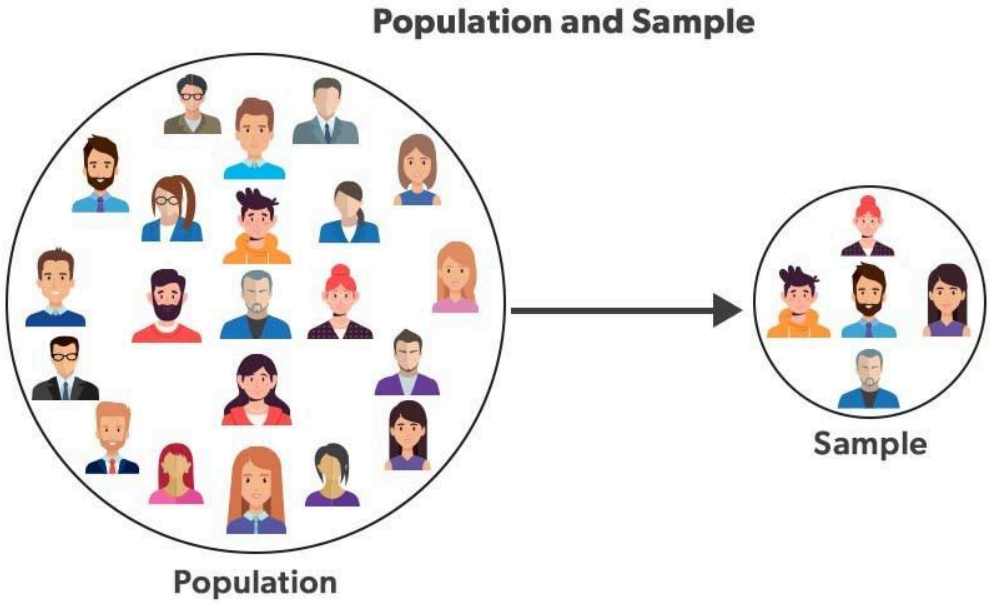
A sample represents a group of the interest of the population which we will use to represent the data. The sample is an unbiased subset of the population in which we represent the whole data. A sample is a group of the elements actually participating in the survey or study.

A sample is the representation of the manageable size. samples are collected and stats are calculated from the sample so one can make interferences or extrapolations from the sample. This process of collecting info from the sample is called sampling.

The sample is denoted by the n

- 500 people from a total population of the Rajasthan state will be considered as a sample
- 143 total chess players from all total number of chess players will be considered as a sample

Sample mean is denoted by x –



Sampling Methods or Types of Sampling

Sample Size: The number of people from the population who are included in the test constitute the sample size. It depends on various factors like the size, the population etc.

Random Sample: When each number/member from the population has an equal chance of being selected. Random Sampling provides us with an unbiased representation of the population.

Example: Picking 10 students randomly from a class of 60.

Types of Random Sampling:

1. Simple Random Sample
2. Systematic Sample
3. Stratified Sample
4. Cluster Sample

Simple Random Sample : When every member of the population has an equal chance of getting selected. In this case the sampling frame includes the whole population. Example: When we use random.sample(range(0,100) ,10) for a population/list of 100 people. It'll select 10 numbers randomly from the list of 100. Every time we rerun this code new 10 samples will be generated every time.

Systematic Sample : This type of sampling is quite similar to Simple random sampling. The difference lies in how we select members from the population, In the case of Systematic sampling we don't randomly select the members but the members are chosen at regular intervals. Example: In a class of 100 students if we select only students whose roll no. is divisible by 10(roll no. are randomly given to each student) is an example of systematic sampling. There is an important thing to note here that there should be no pattern involved during making of the population i.e., if roll numbers are given according to marks achieved in Maths, then this could result in a skewed distribution and there will be a risk that some students might get skipped.

Stratified Sampling: In this technique of sampling, we divide the population into subgroups, these subgroups are often referred to as 'strata'. The division of population into subgroups is not done randomly but based on some similar conditions between the members, for example subgrouping the population based on gender and then taking samples from the subgroups. Now we have subgroups of the main population, to take samples from the subgroups we check the proportion of the original population and then use the same proportion to take samples from the subgroups/stratas. Example: In a class of 100 students 60 are girls and 40 are boys, we divide the population into two subgroups based on gender and then take samples from these two subgroups in 60:40 ratio i.e., if we want to take a sample of 10 students 6 would come from subgroup containing girls and 4 from subgroup containing boys

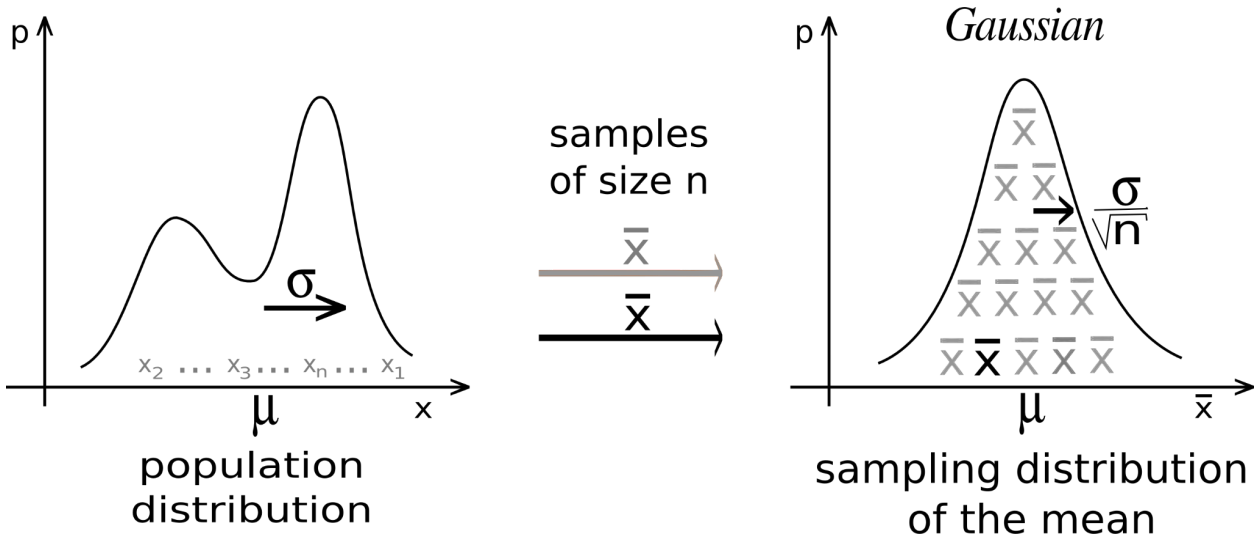
Cluster Sampling: Cluster sampling is similar to Stratified sampling in dividing the population into subgroups. The difference here is that instead of dividing the population into subgroups of distinct features, all subgroups should have similar features to the whole sample. Example: Assume a bank which has offices all over India, now assume they have 100 employees in each state as Managers. If we want to conduct a survey on these managers then we have to collect data across all India that would be a difficult task to perform. Instead of going to every state we just take data from 2-3 states and treat them as our clusters and make our study based on our clusters only. This method is very prone to errors as there can be differences between the data of cities.

Central Limit Theorem

The Central Limit Theorem states that as the sample size grows higher, the sample size of the sampling values approaches a normal distribution, regardless of the form of the data distribution. The mean of sample means will be the population mean, according to the Central Limit Theorem.

Likewise, if you average all the degrees of separation in your sample, you'll get the population's true standard deviation.

- The sample mean is equal to the population mean.
- The sample standard deviation is equal to the population standard deviation divided by the square root of the sample size.



Hypothesis Testing

A hypothesis is an educated guess/claim about a particular property of an object. Hypothesis testing is a way to validate the claim of an experiment.

- **Null Hypothesis :** The null hypothesis is a statement that the value of a population parameter (such as proportion, mean, or standard deviation) is equal to some claimed value. We either reject or fail to reject the null hypothesis. Null Hypothesis is denoted by H0.
- **Alternate Hypothesis :** The alternative hypothesis is the statement that the parameter has a value that is different from the claimed value. It is denoted by HA.

Level of significance : It means the degree of significance in which we accept or reject the null-hypothesis. Since in most of the experiments 100% accuracy is not possible for accepting or rejecting a hypothesis, so we, therefore, select a level of significance. It is denoted by alpha ($\propto$).

Z -Test

Z-test is a statistical method to determine whether the distribution of the test statistics can be approximated by a normal distribution. It is the method to determine whether two sample means are approximately the same or different when their variance is known and the sample size is large (should be >= 30).

When to Use Z-test:

- The sample size should be greater than 30. Otherwise, we should use the t-test.
- Samples should be drawn at random from the population.
- The standard deviation of the population should be known.
- Samples that are drawn from the population should be independent of each other.
- The data should be normally distributed, however for large sample size, it is assumed to have a normal distribution.

$$Z = \frac{(\overline{X} - \mu)}{\left(\sigma / \sqrt{n}\right)}$$

```
data = np.array(np.random.RandomState(8).randint(1, 1000, 500))
```

```
alpha = 0.05  # Significance level
population_mean = np.mean(data)
print('Population Mean :', population_mean)
population_std = np.std(data)
print('Population Std  :', population_std)
```

```
Population Mean : 502.75
Population Std  : 292.8194930328239
```

```
sample_data = np.random.RandomState(8).choice(data, 50)

sample_size = len(sample_data)
print('Sample Size N :', sample_size)

sample_data_mean = np.mean(sample_data)
print('Sample Mean   :', sample_data_mean)
```

```
Sample Size N : 50
Sample Mean   : 491.26
```

```
standard_error = population_std / np.sqrt(sample_size)   # Standard error (standard_error = standard_deviation / square_root(sample_size))
print('Standard error :', standard_error)
```

```
Standard error : 41.41092983742336
```

```
z_score = (sample_data_mean - population_mean) / standard_error
print('z Score :', z_score)

p_value = 2 * (1 - 0.5 * (1 + erf(abs(z_score) / np.sqrt(2))))
print('P Value :', p_value)
```

```
z Score : -0.27746298006610837
P Value : 0.7814246279226427
```

```
if p_value < alpha:
    print("Reject the null hypothesis (The sample mean is significantly different from the population mean).")
else:
    print("Fail to reject the null hypothesis (The sample mean is not significantly different from the population mean).")
```

```
Fail to reject the null hypothesis (The sample mean is not significantly different from the population mean).
```

T - Test

The t-test is named after William Sealy Gosset's Student's t-distribution, created while he was writing under the pen name "Student."

A t-test is a type of inferential statistical test used to determine if there is a significant difference between the means of two groups. It is often used when data is normally distributed and population variance is unknown.

The t-test is used in hypothesis testing to assess whether the observed difference between the means of the two groups is statistically significant or just due to random variation.

Assumptions in T-test

- *Independence :* The observations within each group must be independent of each other. This means that the value of one observation should not influence the value of another observation. Violations of independence can occur with repeated measures, paired data, or clustered data.
- *Normality :* The data within each group should be approximately normally distributed i.e the distribution of the data within each group being compared should resemble a normal (bell-shaped) distribution. This assumption is crucial for small sample sizes ($n < 30$).
- *Homogeneity of Variances (for independent samples t-test) :* The variances of the two groups being compared should be equal. This assumption ensures that the groups have a similar spread of values. Unequal variances can affect the standard error of the difference between means and, consequently, the t-statistic.
- *Absence of Outliers :* There should be no extreme outliers in the data as outliers can disproportionately influence the results, especially when sample sizes are small.

Types of T-tests

There are three types of t-tests, and they are categorized as dependent and independent t-tests.

1. One sample t-test test: The mean of a single group against a known mean.
2. Two-sample t-test: It is further divided into two types:
    a. Independent samples t-test: compares the means for two groups.
    b. Paired sample t-test: compares means from the same group at different times (say, one year apart).

```
pop_mean = 70
sam_mean = 75
sam_std = 27
sam_size = 30
alpha = 0.05
```

```
df = sam_size - 1   # degree of freedom
```

```
t_stat = (sam_mean - pop_mean) / (sam_std / np.sqrt(sam_size))
print('T-Statistic      :', t_stat)

critical_t = stats.t.ppf(1 - alpha, df)
print('Critical T Value :', critical_t)

p_value = 1 - stats.t.cdf(t_stat, df)
print('P Value          :', p_value)
```

```
T-Statistic      : 1.0143010324169741
Critical T Value : 1.6991270265334972
P Value          : 0.15941369166237795
```

```
if t_stat > critical_t :
    print("""There is a significant difference in weight before and after the camp.
    The fitness camp had an effect.""")
else :
    print("""There is no significant difference in weight before and after the camp.
    The fitness camp did not have a significant effect.""")
```

```
There is no significant difference in weight before and after the camp.
    The fitness camp did not have a significant effect.
```

Chi-Square Test

Pearson's Chi-Square is a statistical hypothesis test for independence between categorical variables.

The Contingency Table

The Contingency table (also called crosstab) is used in statistics to summarise the relationship between several categorical variables. Here, we are taking a table that shows the number of men and women buying different types of pets.

	dog	cat	bird	total
men	207	282	241	730
women	234	242	232	708
total	441	524	473	1438

If our calculated value of chi-square is less than or equal to the tabular(also called critical) value of chi-square, then we will accept our Ho.

Chi-square Test for feature selection

Feature selection is also known as attribute selection is a process of extracting the most relevant features from the dataset and then applying machine learning algorithms for the better performance of the model. A large number of irrelevant features increases the training time exponentially and increases the risk of overfitting.

Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi-square scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.

```
from sklearn.datasets import fetch_california_housing, load_iris
from sklearn.feature_selection import SelectKBest, chi2
```

```
data = load_iris()
X = data.data
y = data.target
```

```
model = SelectKBest(score_func = chi2, k = 2)  # k = number of feature we want after selection
```

```
dtf = model.fit_transform(X, y)
```

```
print('Original Features             :', X.shape[1])
print('Features after Chi2 Selection :', dtf.shape[1])
```

```
Original Features             : 4
Features after Chi2 Selection : 2
```

Data Preprocessing

Understanding Data

Understanding the data is the first and most crucial step in the preprocessing pipeline. It helps to identify the structure, nature, and potential issues within the dataset.

Types Of Data :

There are mainly two types of data.

1. Structured Data
2. Unstructured Data

Structured Data

This type of data has a predefined format or schema. This type of data is typically in form rows and columns, where columns represent different features or dimensions and rows represent each data entry.

Examples :

- Spreadsheets (Excel, CSV files).
- Databases (SQL, NoSQL).
- Contains features (columns) and observations (rows).

Unstructured Data

Unlike structured data, unstructured data does not have a predefined format. It includes formats like images, text documents, audio files, and videos. Processing unstructured data often requires more complex techniques such as natural language processing (NLP) or image recognition.

Examples:

- Text (emails, social media posts).
- Images (JPEG, PNG).
- Audio (MP3, WAV).
- Video (MP4, MOV).

Key Terminology

Features :

These are the individual measurable properties or characteristics of the data. In a dataset predicting house prices, features might include the number of bedrooms, square footage, and location. In simple terms it is columns in structured data.

Types:

- Numerical: Continuous (e.g., age, salary) or discrete (e.g., count of items).
- Categorical: Nominal (e.g., color: red, blue, green) or ordinal (e.g., education level: high school, bachelor's, master's).
- Date/Time: Timestamps or durations (e.g., year, month, day).

Target Variable :

This is the outcome variable that you want to predict or classify. In the house price example, the target variable would be the price of the house.

Examples:

- *Regression:* House prices, stock prices (continuous values).
- *Classification:* Spam or not spam, dog or cat.

Training/Testing Data :

In Machine Learning and Deep Learning data is generally divided in 2 parts, training(and or validation data) and testing for model evaluation.

Training Data : It is the subset of the original data that is used to train the ml model. Primary purpose of it is to learn complex patterns and relationships in the dataset.

Testing Data : Another subset of original data which is not seen by ml model. Primary purpose of this data is to evaluate the models performance and generalization capabilities.

Validation Data : A portion of the training data set aside to tune hyperparameters. Primary purpose is to help in model selection and avoid overfitting.

Understanding Dataset Dimensions

a) Rows (Observations/Instances):

- Each row corresponds to an individual data point.
- Example: A row in a housing dataset might represent one house.

b) Columns (Features/Attributes):

- Each column represents a characteristic or property of the data.
- Example: Columns like "square footage," "number of bedrooms," and "price" in a housing dataset.

Data Cleaning

Data cleaning is a vital step in preprocessing, ensuring that the dataset is accurate, consistent, and ready for analysis or modeling.

Handling Missing Values

Drop Rows/Columns : If a significant amount of data is missing from a row or column, it may be more effective to remove it entirely.

```
df2.drop(['last_review', 'reviews_per_month'], axis = 1, inplace = True)  # droping columns that have null values
```

```
df2.isnull().sum()
```

```
id                                0
name                             16
host_id                           0
host_name                        21
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

```
df3.dropna(inplace = True)   # droping rows that have null values
```

```
df3.isnull().sum()
```

```
id                                0
name                              0
host_id                           0
host_name                         0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

Mean/Median/Mode Imputation : Replace missing values with the mean (average), median (middle value), or mode (most frequent value) of the feature.

```
reviews_per_month_mean = df['reviews_per_month'].mean()
```

```
df4['reviews_per_month'] = df4['reviews_per_month'].fillna(reviews_per_month_mean)
```

```
df4.isnull().sum()
```

```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                     0
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

```
df5['host_name'] = df5['host_name'].fillna(df5['host_name'].value_counts().index[0])
```

```
df5.isnull().sum()
```

```
id                                    0
name                                 16
host_id                               0
host_name                             0
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

```
df6['last_review'] = df['last_review'].fillna('Not_Reviewed')
```

```
df6.isnull().sum()
```

```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                           0
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

```
df7['reviews_per_month'] = df7['reviews_per_month'].fillna(df7['reviews_per_month'].median())
```

```
df7.isnull().sum()
```

```
id                                  0
name                               16
host_id                             0
host_name                          21
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month                   0
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

```
df8['reviews_per_month'] = df8['reviews_per_month'].interpolate()
```

```
df8.isnull().sum()
```

```
id                                  0
name                               16
host_id                             0
host_name                          21
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month                   0
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

Removing Duplicates

Duplicate records can skew analysis and model training. Identifying and removing duplicates ensures that each observation is unique.

```
df9 = df.copy()
```

```
df9.duplicated().sum()
```

```
11
```

```
df9 = df9.drop_duplicates()
```

```
df9.duplicated().sum()
```

```
0
```

Handling Outliers

Outliers can distort statistical analyses and model performance. Common methods for identifying outliers include:

- *Interquartile Range (IQR) :* Calculate Q1 (25th percentile) and Q3 (75th percentile) to find IQR = Q3 - Q1. Outliers are typically defined as values below Q1 - 1.5IQR or above Q3 + 1.5IQR.
- *Z-Score :* Calculate the z-score for each value in a feature; values with z-scores greater than 3 or less than -3 are often considered outliers.

```
Highest_allowed = df2["semester_percentage"].mean() + 3 * df2["semester_percentage"].std()
Lowest_allowed = df2["semester_percentage"].mean() - 3 * df2["semester_percentage"].std()
print("Highest allowed :", Highest_allowed)
print("Lowest allowed  :", Lowest_allowed)
```

```
Highest allowed : 88.08933625397168
Lowest allowed  : 51.13546374602831
```

```
print("Outliers are showed below")
df2[(df2["semester_percentage"] < Lowest_allowed) | (df2["semester_percentage"] > Highest_allowed) ]
```

```
Outliers are showed below
```

	semester_percentage	scholarship_exam_marks	got_scholarship
485	49.2	44	1
995	88.7	44	1
996	91.2	65	1
997	48.9	34	0
999	49.0	10	1

```
Remove that outliers
```

```
df2[(df2["semester_percentage"] > Lowest_allowed) & (df2["semester_percentage"] < Highest_allowed) ]
```

	semester_percentage	scholarship_exam_marks	got_scholarship
0	71.9	26	1
1	74.6	38	1
2	75.4	40	1
3	64.2	8	1
4	72.3	17	0
...	...	...	...
991	70.4	57	0
992	62.6	12	0
993	67.3	21	1
994	64.8	63	0
998	86.2	46	1

```
df2["z_score"] = (df2["semester_percentage"] - df2["semester_percentage"].mean()) / df2["semester_percentage"].std()
```

```
df2.head()
```

	semester_percentage	scholarship_exam_marks	got_scholarship	z_score
0	71.9	26	1	0.371425
1	74.6	38	1	0.809810
2	75.4	40	1	0.939701
3	64.2	8	1	-0.878782
4	72.3	17	0	0.436371

```
df2[(df2["z_score"] > -3) & (df2["z_score"] < 3)]
```

	semester_percentage	scholarship_exam_marks	got_scholarship	z_score
0	71.9	26	1	0.371425
1	74.6	38	1	0.809810
2	75.4	40	1	0.939701
3	64.2	8	1	-0.878782
4	72.3	17	0	0.436371
...	...	...	...	...
991	70.4	57	0	0.127878
992	62.6	12	0	-1.138565
993	67.3	21	1	-0.375452
994	64.8	63	0	-0.781363
998	86.2	46	1	2.693239

995 rows × 4 columns

```
upper_limit = df2["semester_percentage"].mean() + 3 * df2["semester_percentage"].std()
lower_limit = df2["semester_percentage"].mean() - 3 * df2["semester_percentage"].std()
```

```
df2['semester_percentage'] = np.where(df2['semester_percentage'] > upper_limit, upper_limit,
                                      np.where(df2['semester_percentage'] < lower_limit, lower_limit,
                                      df2['semester_percentage']))
```

```
df2[df2['semester_percentage'] > upper_limit]
```

	semester_percentage	scholarship_exam_marks	got_scholarship	z_score

```
df2[df2['semester_percentage'] < lower_limit]
```

	semester_percentage	scholarship_exam_marks	got_scholarship	z_score