Statistics with Python

Morteza Mohammadi

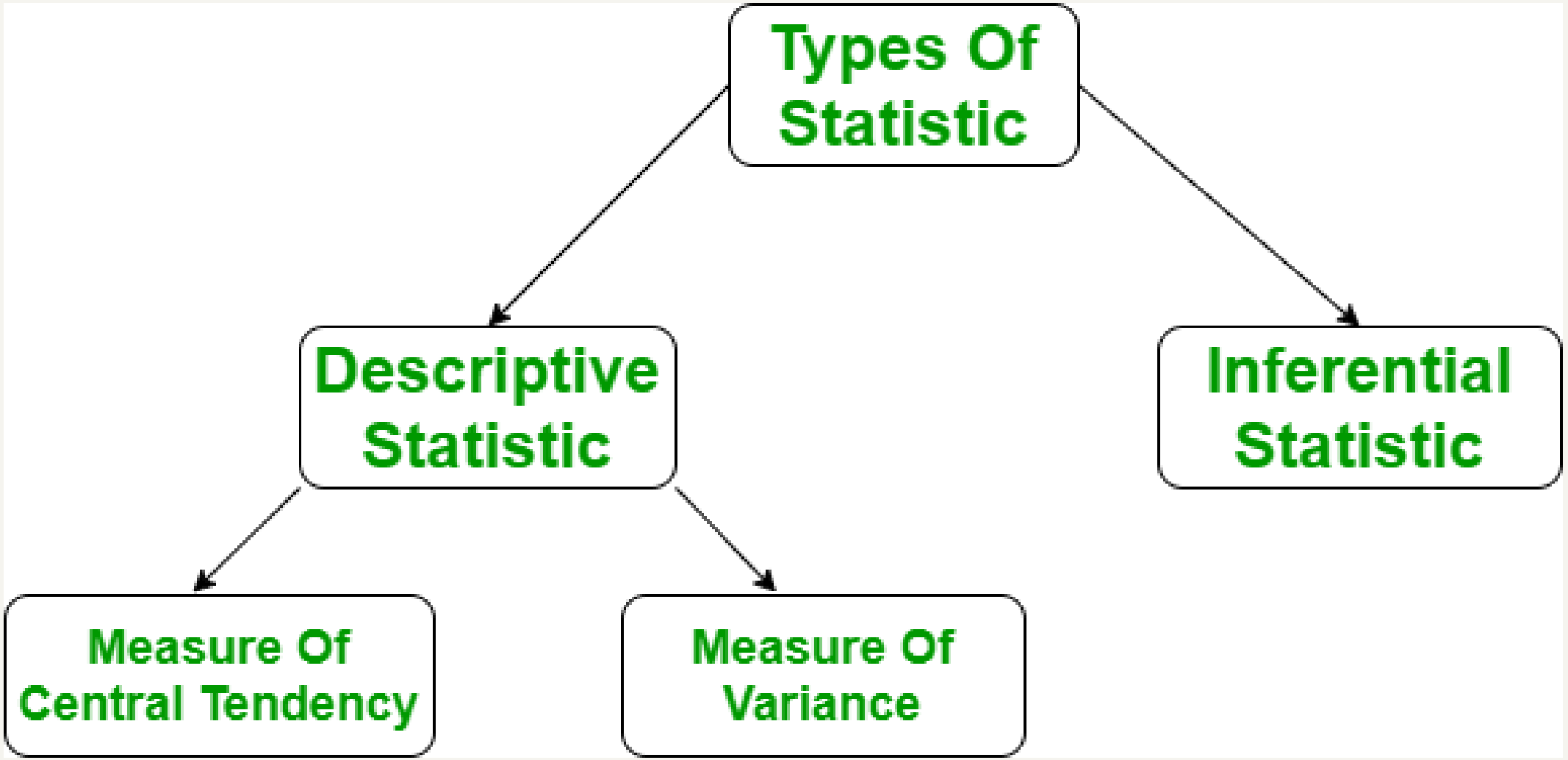
Department of Statistics

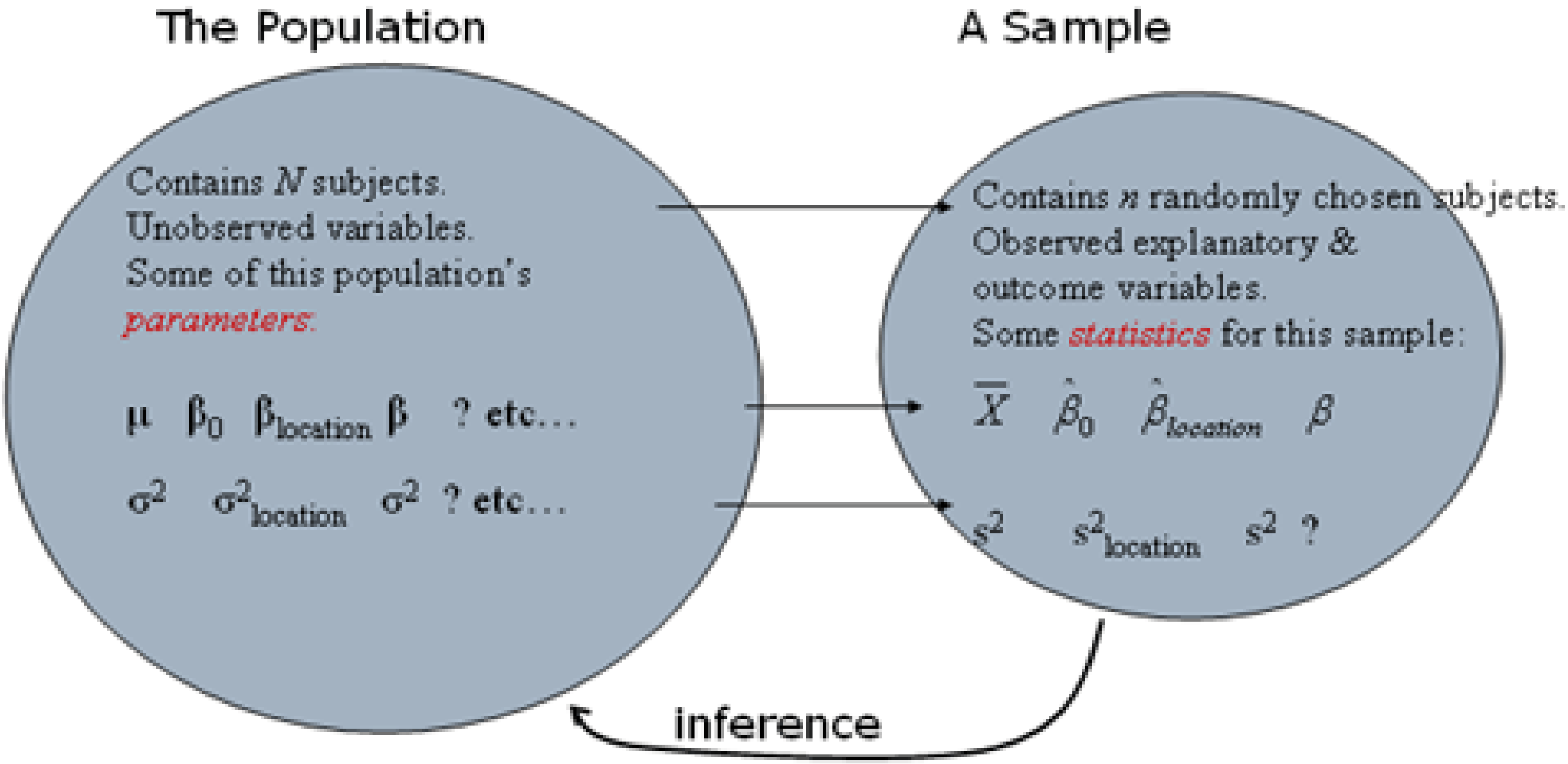
University of Zabol

Python Statistics Fundamentals: How to Describe Your Data – Real Python

Statistics

- The science of collecting, analyzing, presenting, and interpreting data.


Types Of Statistic
Descriptive Statistic
Inferential Statistic
Measure Of Central Tendency
Measure Of Variance



μ = true unknown average price of 1bd in SC

$\overline{X}$ = estimated sample average price of 1bd apartment in SC

Probability distribution

A **probability distribution** is the mathematical function that gives the probabilities of occurrence of different possible **outcomes** for an experiment.

Three important distributions:

1. Normal Distribution
2. Student T Distribution
3. F (Fisher) Distribution

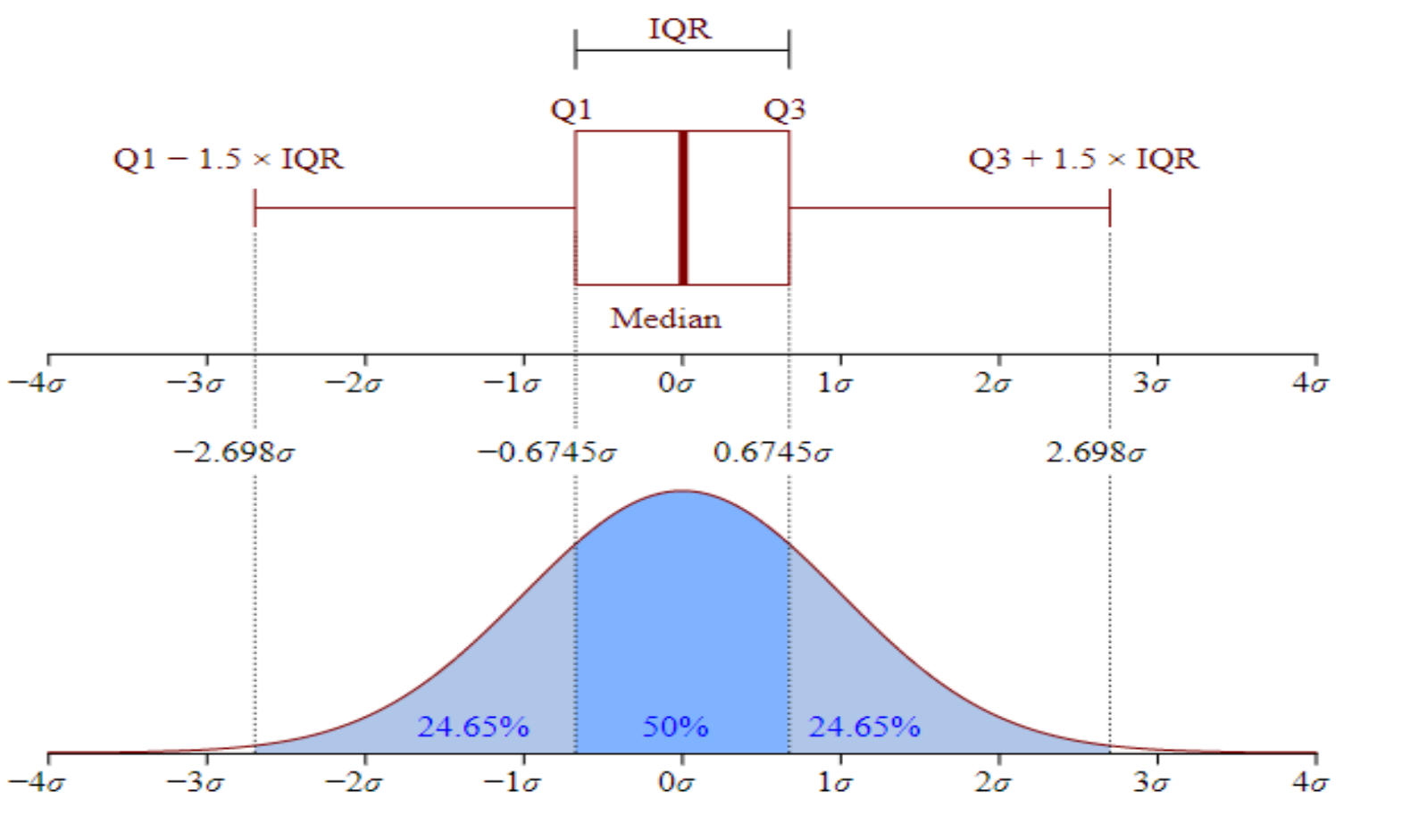
Normal Distribution

The normal distribution is a continuous probability distribution function also known as Gaussian distribution which is symmetric about its mean and has a bell-shaped curve. It is one of the most used probability distributions. Two parameters characterize it

- Mean(μ)- It represents the center of the distribution
- Standard Deviation(σ) – It represents the spread in the curve

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

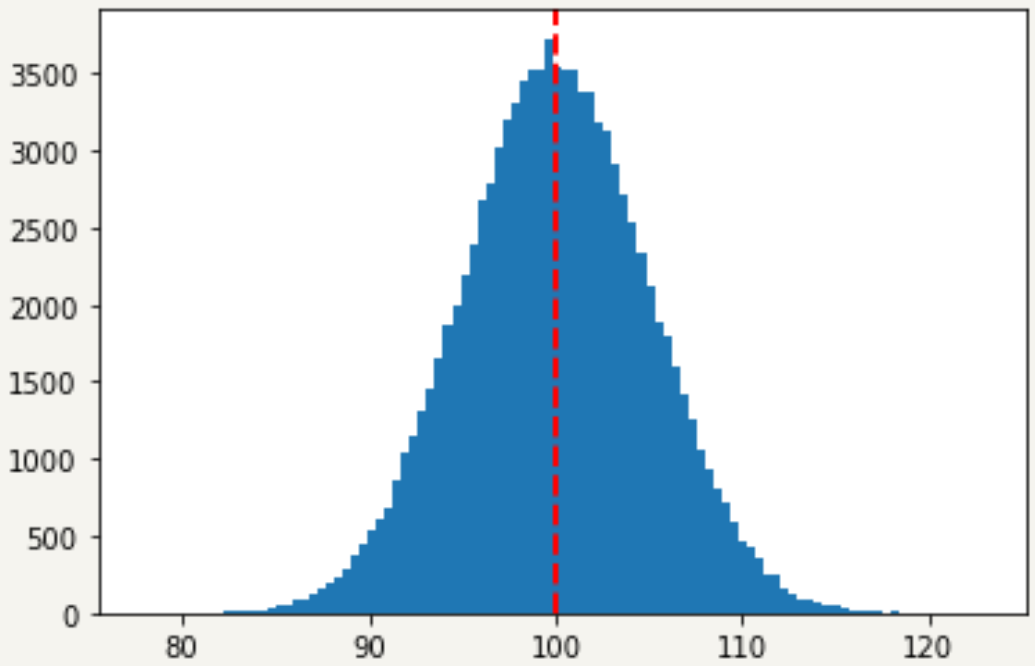
Normal Distribution



Normal Distribution Using Python

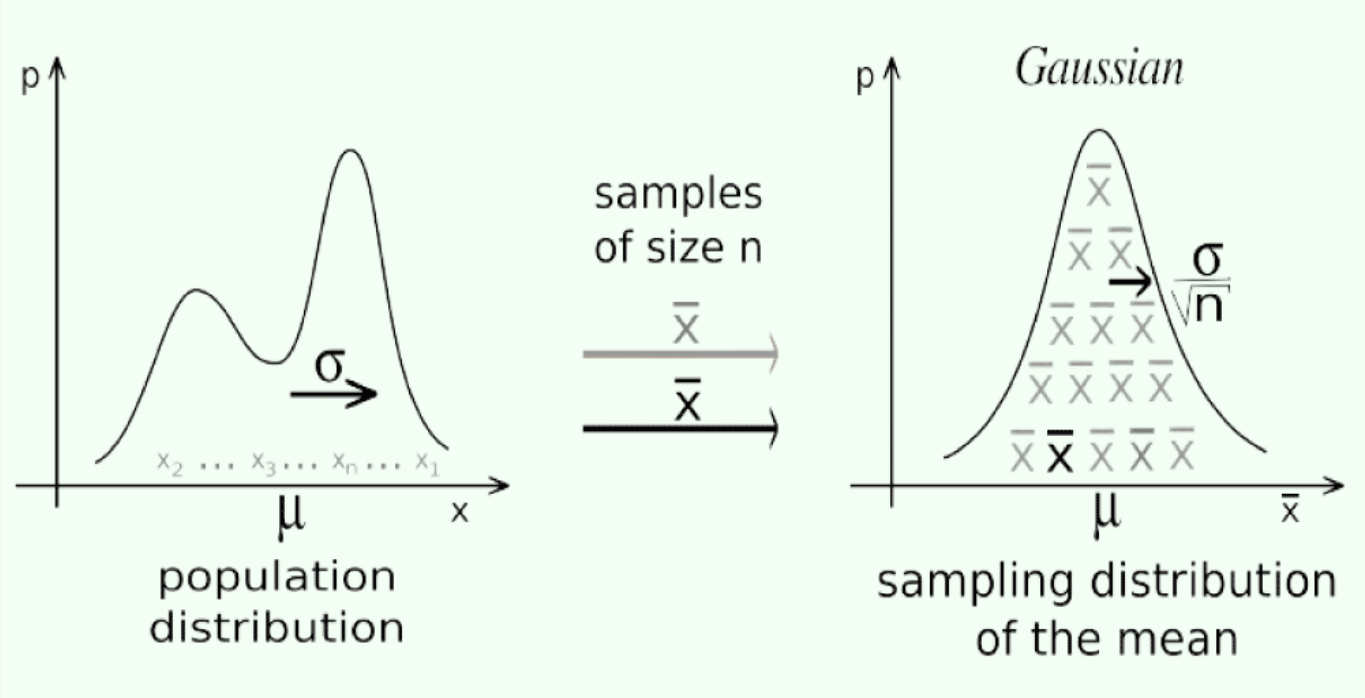
```
>>> import numpy as np
>>> def normal_dist(x, mean, sd):
        prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)
        return prob_density

>>> mean = 0
>>> sd = 1
>>> x = 1
>>> normal_dist(x, mean, sd)
1.9054722647301798
```

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> Mean = 100
>>> Standard_deviation = 5
>>> size = 100000
>>> values = np.random.normal(Mean, Standard_deviation, size)
>>> plt.hist(values, 100)
>>> plt.axvline(values.mean(), color='red', linestyle='dashed', linewidth=2)
plt.show()
```



Central limit theoram

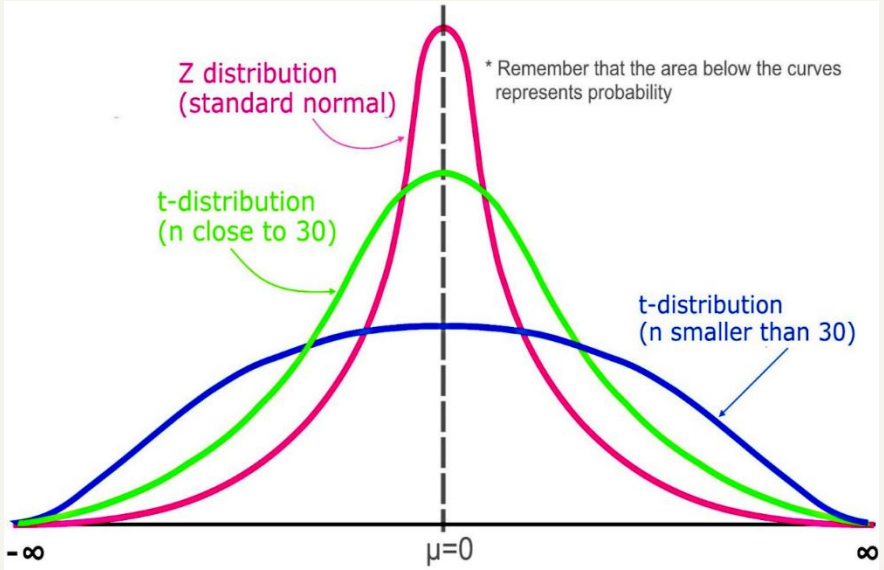
Generally, the Central Limit Theoram is used when the sample size is fairly big, usually larger than or equal to 30. In some cases even if the sample size is less than 30 central limit theoram still holds but for this the population distribution should be close to normal or symmetric.



T-distribution

- The t-distribution is used in statistics to estimate the significance of population parameters for small sample sizes or unknown variations. Like the normal distribution, it is bell-shaped and symmetric. Unlike normal distributions, it has heavier tails, which result in a greater chance for extreme values.

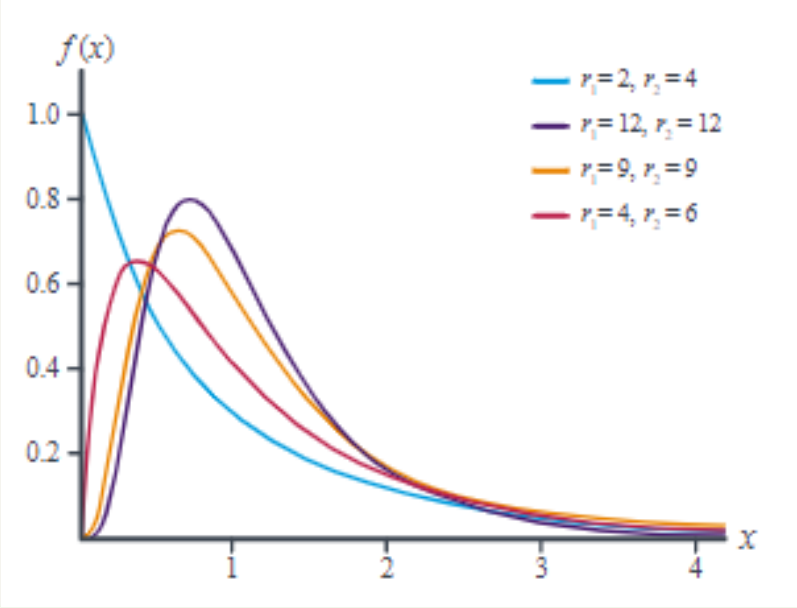
$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{t^2}{\nu}\right)^{-(\nu+1)/2}$$



F-Distribution

- *F*-distributions are generally skewed. The shape of an *F*-distribution depends on the values of r1 and r2, the numerator and denominator degrees of freedom, respectively,

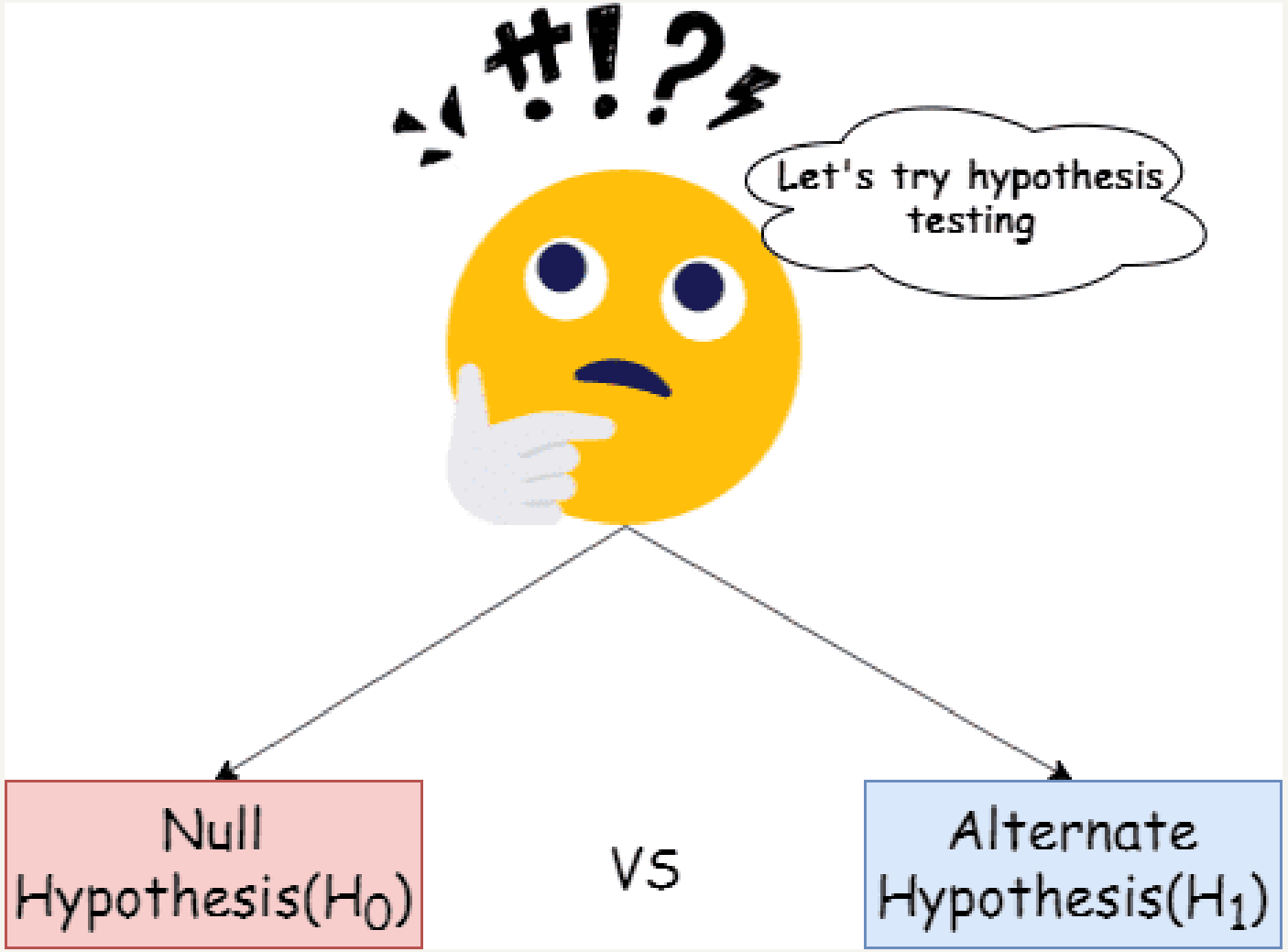
$$f(w) = \frac{(r_1/r_2)^{r_1/2}\Gamma[(r_1+r_2)/2]w^{(r_1/2)-1}}{\Gamma[r_1/2]\Gamma[r_2/2][1+(r_1 w/r_2)]^{(r_1+r_2)/2}}$$



Hypothesis Testing

- Hypothesis testing is a statistical method that is used in making a statistical decision using experimental data. Hypothesis testing is basically an assumption that we make about a population parameter.

Example: You say an average student in the class is 30 or a boy is taller than a girl.


Let's try hypothesis testing
Null Hypothesis(H_0)
VS
Alternate Hypothesis(H_1)

Decision Matrix

		RESEARCHER'S CONCLUSION	
		Fail to reject H_0	Reject H_0
REALITY	H_0 is true		Type I error α
	H_0 is false	Type II error β	

Hypotheses for One Sample Test

The null hypothesis (H_0) and (two-tailed) alternative hypothesis (H_1) of the one sample T test can be expressed as:

$$H_0: \mu = \mu_0$$

("the population mean is equal to the [proposed] population mean")

$$H_1: \mu \neq \mu_0$$

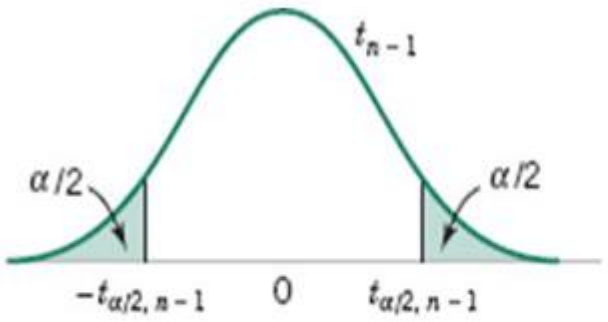
("the population mean is not equal to the [proposed] population mean")

where μ is the "true" population mean and μ_0 is the proposed value of the population mean.

Hypothesis Testing: One Population

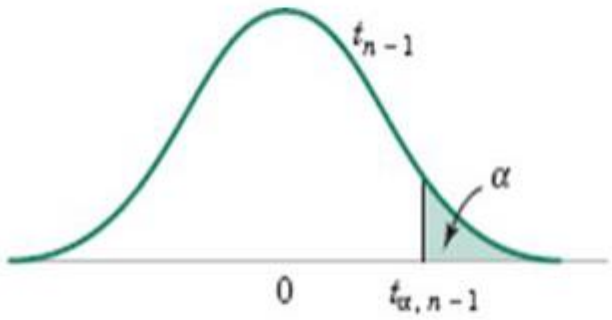
Hypothesis Tests for the Population Mean

Null Hypothesis (Ho)	*Alternative Hypothesis (Ha)*	*Test Statistic*	*Rejection Region*
Case 1: σ^2 is known $\mu = \mu_o$	$\mu < \mu_o$ $\mu > \mu_o$ $\mu \neq \mu_o$	$Z = \frac{\bar{X} - \mu_o}{\sigma / \sqrt{n}}$	$z < -z_\alpha$ $z > z_\alpha$ $\lvert z \rvert > z_{\frac{\alpha}{2}}$
Case 2: σ^2 is unknown and n ≤ 30 $\mu = \mu_o$	$\mu < \mu_o$ $\mu > \mu_o$ $\mu \neq \mu_o$	$T = \frac{\bar{X} - \mu_o}{s / \sqrt{n}}$	$t < -t_{\alpha, n-1}$ $t > t_{\alpha, n-1}$ $\lvert t \rvert > t_{\frac{\alpha}{2}, n-1}$
Case 3: σ^2 is unknown and n > 30 $\mu = \mu_o$	$\mu < \mu_o$ $\mu > \mu_o$ $\mu \neq \mu_o$	$Z = \frac{\bar{X} - \mu_o}{s / \sqrt{n}}$	$z < -z_\alpha$ $z > z_\alpha$ $\lvert z \rvert > z_{\frac{\alpha}{2}}$



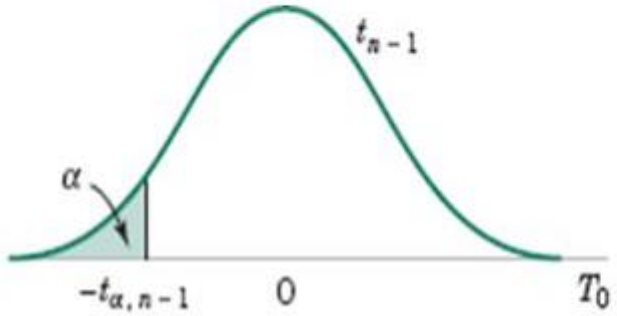
- Two tail test:

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu \neq \mu_0$$



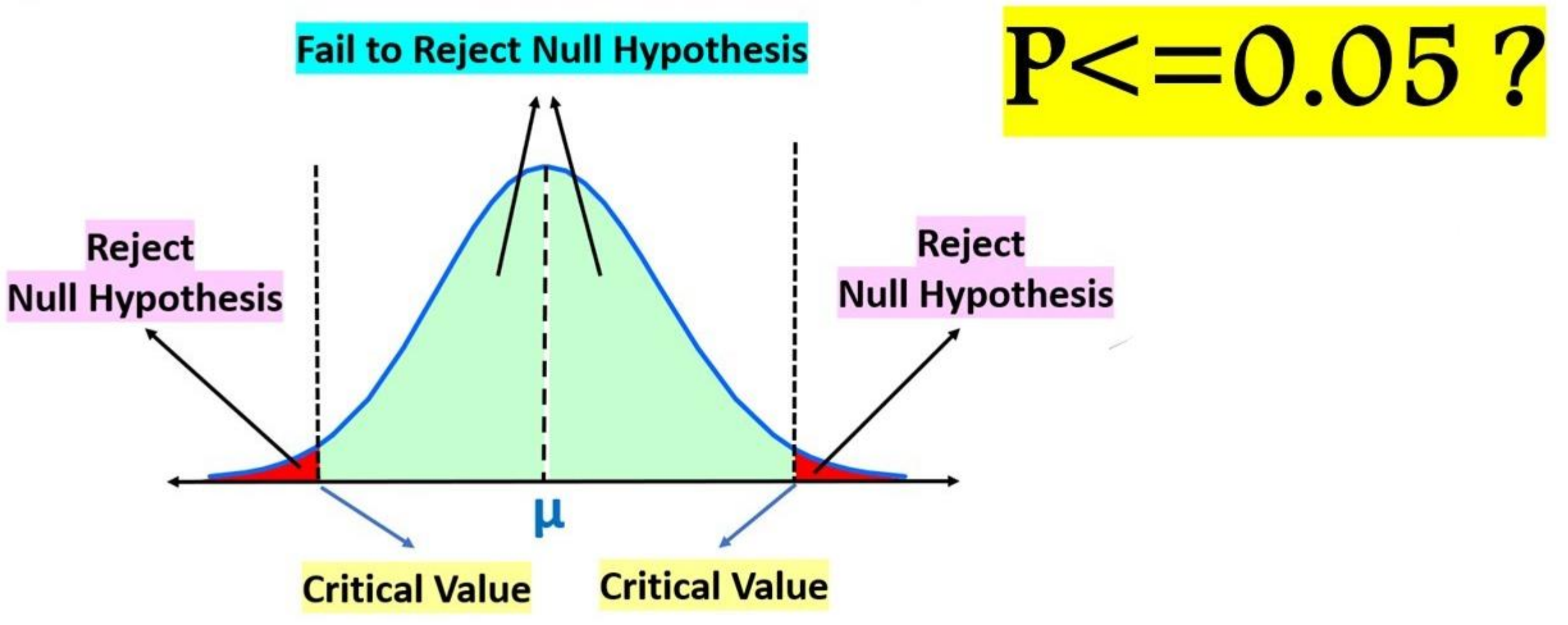
- Upper tail test

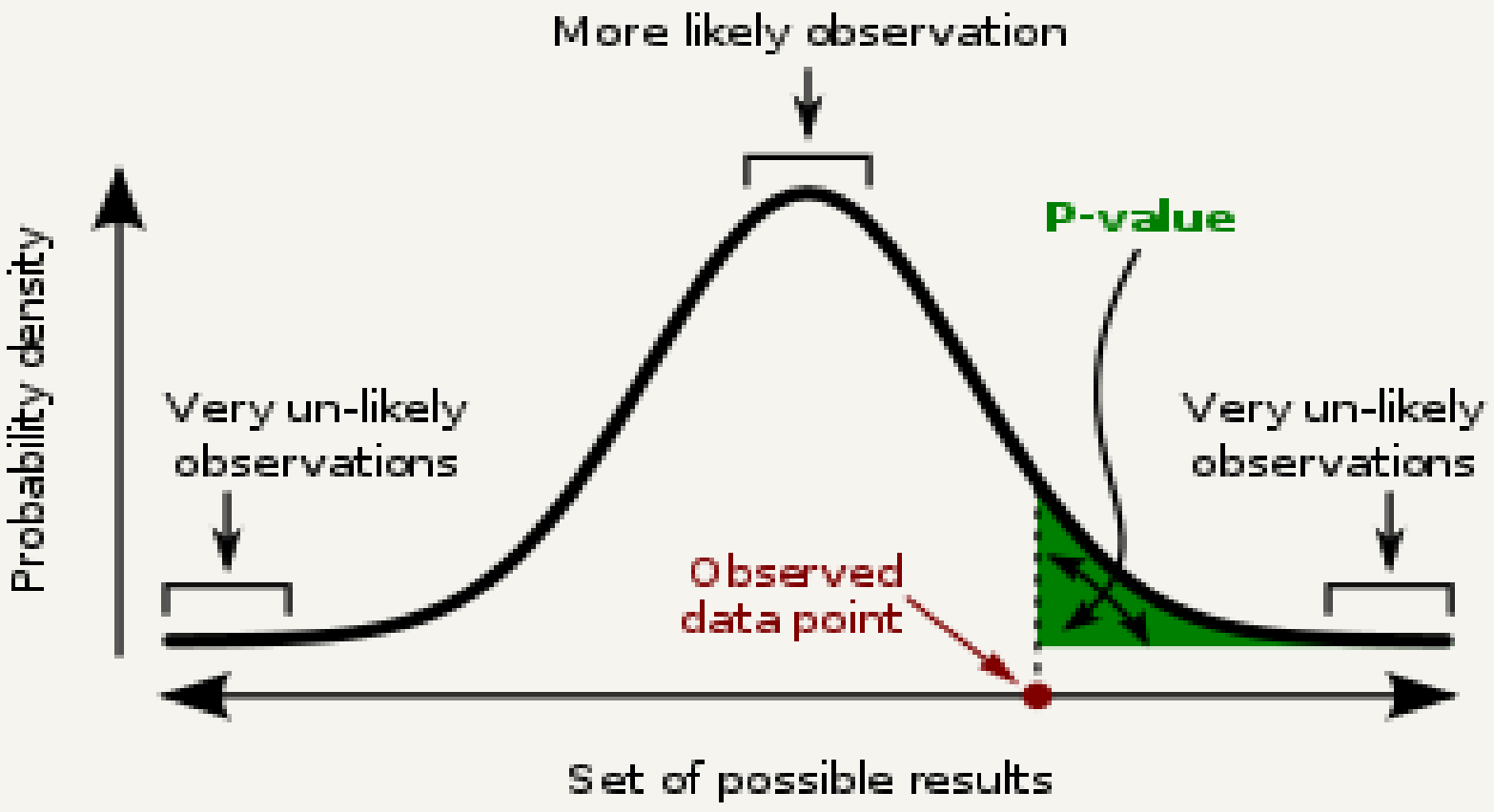
$$H_0 : \mu \leq \mu_0$$
$$H_1 : \mu > \mu_0$$



- Lower tail test

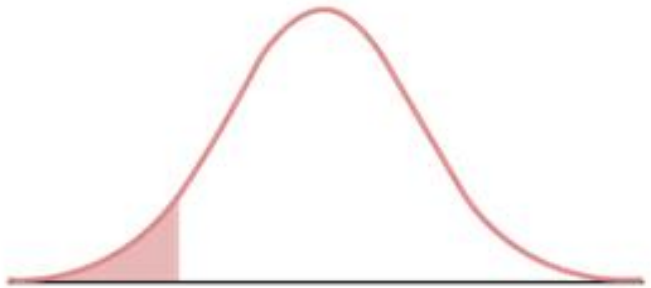
$$H_0 : \mu \geq \mu_0$$
$$H_1 : \mu < \mu_0$$


Fail to Reject Null Hypothesis
P<=0.05 ?
Reject
Null Hypothesis
Reject
Null Hypothesis
μ
Critical Value
Critical Value



A **p-value** (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

If $\mu < claim$, then $p-value = P(t < test\ statistic)$



(lower) one-tailed test

If $\mu > claim$, then $p-value = P(t > test\ statistic)$

(upper) one-tailed test

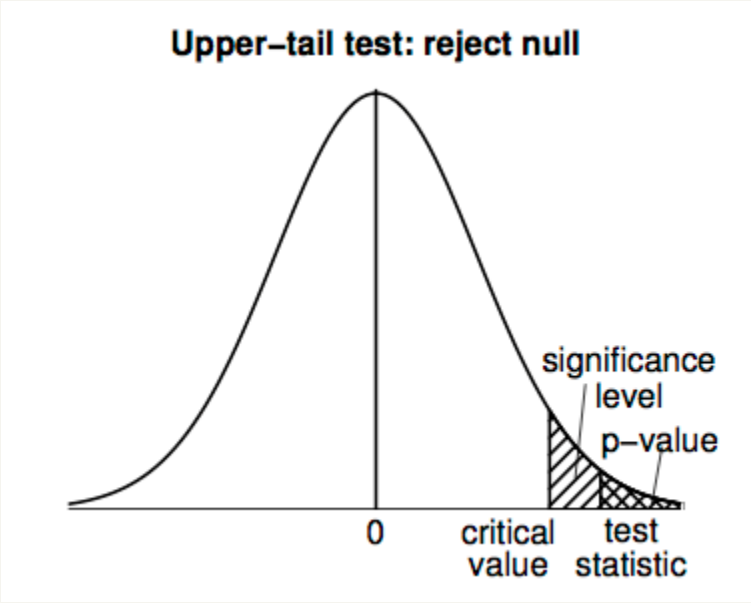
If $\mu \neq claim$, then $p-value = 2P(t > |test\ statistic|)$

two-tailed test

P-Value Interpretation:

If $p-value \leq significance\ level$, then we <u>reject</u> H_0

If $p-value > significance\ level$, then we <u>fail to reject</u> H_0



Procedure

1. Calculate the test statistic
2. Determine our p-value
3. Compare
   - If our p-value is less than or equal to our significance level, we will reject our null hypothesis.
   - Otherwise we fail to reject the null hypothesis

Example:

For example, imagine a company wants to test the claim that their batteries last more than 40 hours. Using a simple random sample of 15 batteries yielded a mean of 44.9 hours, with a standard deviation of 8.9 hours. Test this claim using a significance level of 0.05.

$$H_0: \mu = 40$$
$$H_a: \mu > 40$$

$$\hat{x} = 44.9, \quad \mu = 40 \quad s = 8.9, \quad n = 15, \quad df = n - 1 \rightarrow df = 15 - 1 = 14$$

$$test\ statistic: \quad t = \frac{44.9 - 40}{\left(\frac{8.9}{\sqrt{15}}\right)} = 2.13$$

$$p - value = P\left(t_{df=14} > 2.13\right) = 0.026$$

Because $p = 0.026 < \alpha = 0.05$ we reject H_0

Python

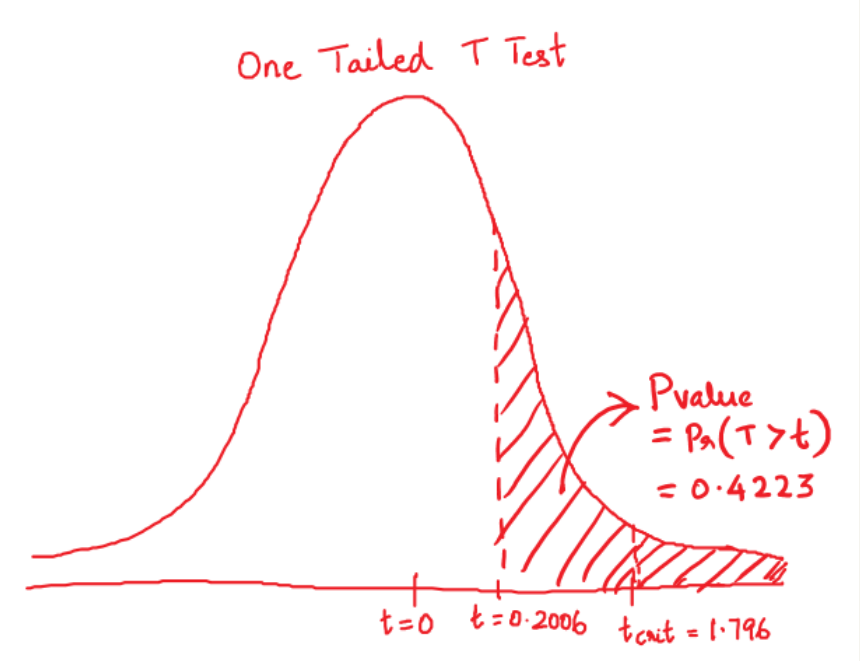
- In Python, One sample T Test is implemented in ttest_1samp() function in the scipy package. However, it does a Two tailed test by default, and reports a signed T statistic. That means, the reported P-value will always be computed for a Two-tailed test. To calculate the correct P value, you need to divide the output P-value by 2.

Apply the following logic if you are performing a one tailed test:

- For greater than test: Reject H0 if p/2 < alpha (0.05). In this case, t will be greater than 0.
- For lesser than test: Reject H0 if p/2 < alpha (0.05). In this case, t will be less than 0.

```
>>> from scipy.stats import ttest_1samp
>>> x =  [21.5, 24.5, 18.5, 17.2, 14.5, 23.2, 22.1, 20.5, 19.4, 18.1, 24.1, 18.5]
>>> tscore, pvalue = ttest_1samp(x, popmean=20)
>>> print("t Statistic: ", tscore)
>>> print("P Value: ", pvalue)
#> t Statistic:  0.2006562773994862
#> P Value:  0.8446291893053613
```

$$H_0: \mu = 20$$
$$H_1: \mu > 20$$



Since it is one tailed test, the real p-value is 0.8446/2 = 0.4223. We do not rejecting the Null Hypothesis anyway.

Paired Samples t Test

- The Paired Samples t Test compares the means of two measurements taken from the same individual, object, or related units. These "paired" measurements can represent things like: A measurement taken at two different times (e.g., pre-test and post-test score with an intervention administered between the two time points)

$$H_0: \mu_{pre} = \mu_{post}$$
$$H_1: \mu_{pre} \neq \mu_{post}$$

$$t = \frac{\bar{x}_d - \mu_d}{\left(\frac{s_d}{\sqrt{n}}\right)} \quad , \quad df = n - 1$$

$\bar{x}_d$: sample mean difference

μ : population mean difference

s : sample difference standard deviation

n : sample size

```
#two related or repeated samples
>>> import scipy.stats as stats
# pre score
>>> pre = [88, 82, 84, 93, 75, 78, 84, 87, 95, 91, 83, 89, 77, 68, 91]
# post score
>>> post = [91, 84, 88, 90, 79, 80, 88, 90, 90, 96, 88, 89, 81, 74, 92]
stats.ttest_rel(pre, post)
Ttest_relResult(statistic=-2.9732484231168, pvalue=0.010071448626 4)
```

Independent samples t-test

$$H_0: \mu_1 = \mu_2$$
$$H_1: \mu_1 \neq \mu_2$$

- a two independent samples t-test, also known as unpaired two samples t-test, is an essential statistical tool that can help you draw meaningful conclusions from your data. This test allows you to determine whether the difference between the means of two independent samples is statistically significant or due to chance.

Id	Gender	Mathematics Marks
1	Male	98
2	Male	92
3	Male	89
4	Male	75
5	Female	83
6	Female	92
7	Female	85
8	Female	99

An assumption for independent samples t-test is **homogeneity of variance of the two groups**.

Two Sample Test

Two Sample Means

Test Statistic when σ known

$$z = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_2 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$F = \frac{s_1^2}{s_2^2} = \frac{larger\ variance}{smaller\ variance}$$

If $F \le t^*_{(\alpha, df)}$, then pool

If $F > t^*_{(\alpha, df)}$, then don't pool

Test Statistic when σ unknown
With **pooled** variances

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_2 - \mu_2)}{\left(\sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}\right)\left(\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\right)}$$

$$df = n_1 + n_2 - 2$$

Test Statistic when σ unknown
with **un-pooled** variances

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_2 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\left(\frac{1}{n_1 - 1}\right)\left(\frac{s_1^2}{n_1}\right)^2 + \left(\frac{1}{n_2 - 1}\right)\left(\frac{s_2^2}{n_2}\right)^2}$$

```
>>> import numpy as np
>>> from scipy import stats
# Observations for class 1
>>> class1 = [12.5, 11.2, 13.1, 10.8, 11.9, 10.5, 12.4, 12.9, 11.7, 13.2]
# Observations for class 2
>>> class2 = [14.3, 13.1, 15.2, 12.7, 13.9, 13.5, 14.1, 12.8, 13.7, 15.5]
# Perform the two independent samples t-test
>>> stats.ttest_ind(class1, class2)
Ttest_indResult(statistic=-4.379738509717699, pvalue=0.0003613013062951915)
```

ANOVA

- Analysis of Variance (ANOVA) is a statistical formula used to compare variances across the means (or average) of different groups.

$$H_0: \mu_1 = \mu_2 = \cdots = \mu_p$$

$$H_1: \sim H_0$$

The ANOVA Table for Comparing Means

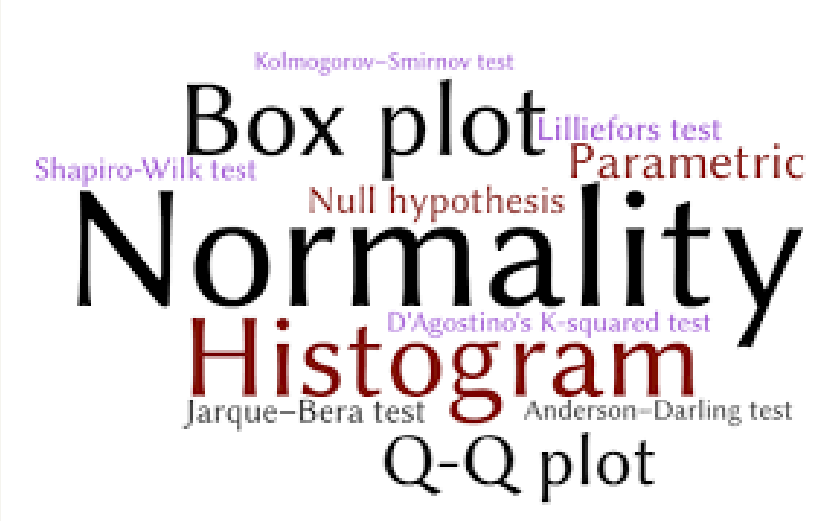
Source	SS *(Sum of Squares, the numerator of the variance)*	DF *(the denominator)*	MS *(Mean Square, the variance)*	F
Treatment *(or Between or Model)*	$SST = \sum_{i=1}^{p} \sum_{j=1}^{n_i} (\bar{y}_i - \bar{y})^2$	*p-1*	$MST = \frac{SST}{p-1}$	$F = \frac{MST}{MSE}$
Error *(or Within)*	$SSE = \sum_{i=1}^{p} \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2$	*n-p*	$MSE = \frac{SSE}{n-p}$	
Total	$TSS = \sum_{i=1}^{p} \sum_{j=1}^{n_i} (y_{ij} - \bar{y})^2$	*n-1*		

```
>>> from scipy.stats import f_oneway

# Performance when each of the engine oil is applied
>>> performance1 = [89, 89, 88, 78, 79]
>>> performance2 = [93, 92, 94, 89, 88]
>>> performance3 = [89, 88, 89, 93, 90]
>>> performance4 = [81, 78, 81, 92, 82]

# Conduct the one-way ANOVA
>>> f_oneway(performance1, performance2, performance3, performance4)
F_onewayResult(statistic=4.625000000000002, pvalue=0.01633645983978)
```

Assumptions

For a valid test analysis, data values must be all of the following:

1. Continuous measurements
2. Normally distributed in the population



Kolmogorov-Smirnov test for goodness of fit

- Suppose we wish to test the null hypothesis that a sample is distributed according to the standard normal.
- we will reject the null hypothesis in favor of the alternative if the p-value is less than 0.05.

```
>>> import numpy as np
>>> from scipy import stats
>>> stats.kstest(np.random.normal(0,1,10000),'norm')
KstestResult(statistic= 0.007038739782416259, pvalue= 0.70477679457831155)

>>> stats.kstest(stats.uniform.rvs(size=100), stats.norm.cdf)
KstestResult(statistic=0.5047799891120425, pvalue=4.046245896284498e-24)
```

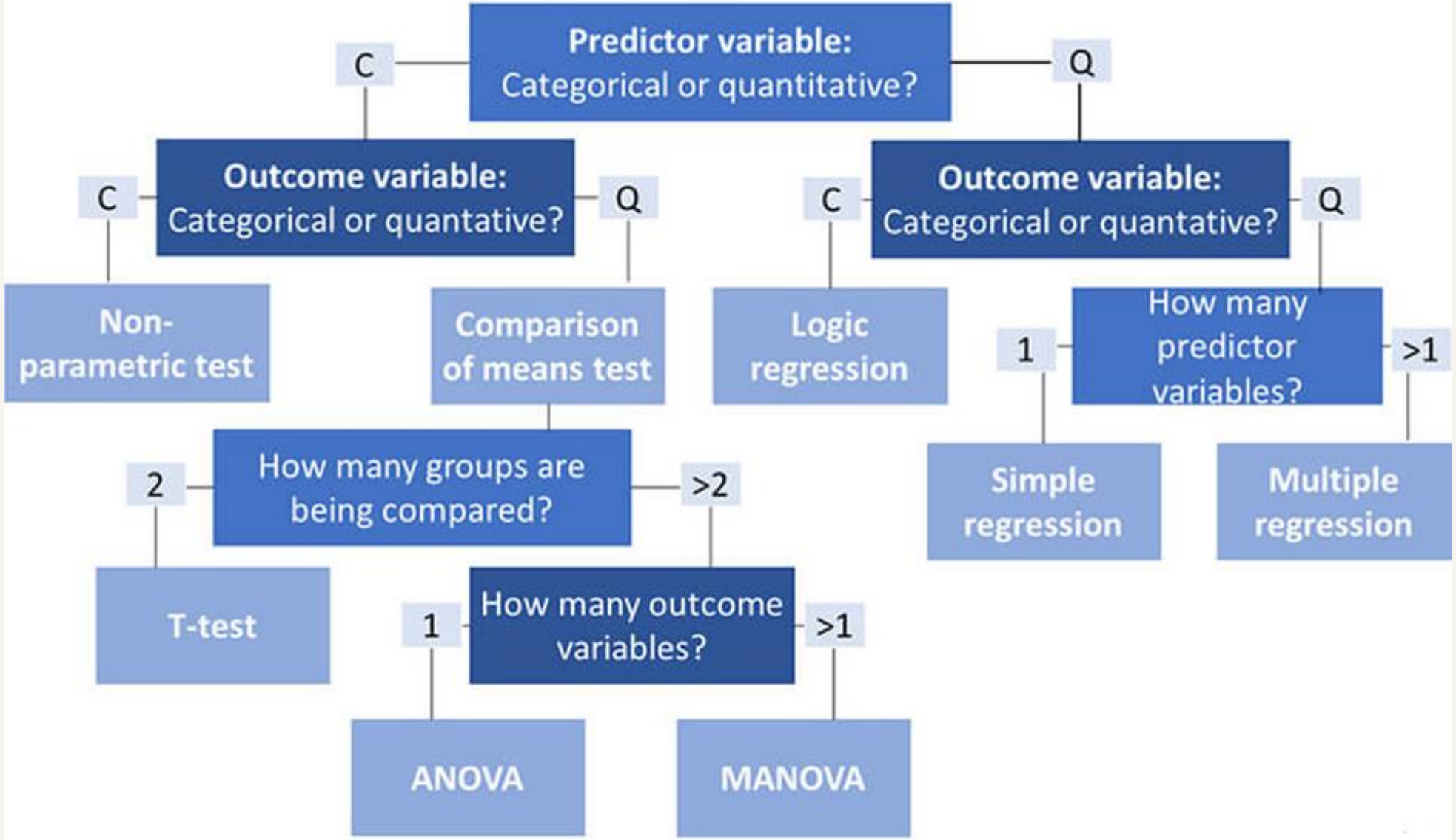
Data		1 Sample	2 Samples	>2 Samples
Continuous & Meets Normality Assumption	Mean	$H_0: \mu = \mu_0$ $H_A: \mu \neq \mu_0$ One Sample T-Test	$H_0: \mu_1 = \mu_2$ $H_A: \mu_1 \neq \mu_2$ Two Sample T-Test	$H_0: \mu_1 = \mu_2 = \ldots = \mu_k$ $H_A:$ At least one mean is different Analysis of Variance
	Standard Deviation	$H_0: \sigma = \sigma_0$ $H_A: \sigma \neq \sigma_0$ Chi-Square Test	$H_0: \sigma_1 = \sigma_2$ $H_A: \sigma_1 \neq \sigma_2$ F-Test	$H_0: \sigma_1 = \sigma_2 = \ldots = \sigma_k$ $H_A:$ At least one st. dev is different Bartlett's Test
Continuous & Non Normal	Median	$H_0: \eta = \eta_0$ $H_A: \eta \neq \eta_0$ Wilcoxon Test	$H_0: \eta_1 = \eta_2$ $H_A: \eta_1 \neq \eta_2$ Mann-Whitney Test	$H_0: \eta_1 = \eta_2 = \ldots = \eta_k$ $H_A:$ At least one mean is different Kruskal-Wallis Test
Discrete	Proportion	$H_0: \Pi = \Pi_0$ $H_A: \Pi \neq \Pi_0$ Test for One Proportion	$H_0: \Pi_1 = \Pi_2$ $H_A: \Pi_1 \neq \Pi_2$ Test for Two Proportions	$H_0: \Pi_1 = \Pi_2 = \ldots = \Pi_k$ $H_A:$ At least one proportion is different Binomial Analysis of Means

H_0 = Null Hypothesis

H_A = Alternative Hypothesis

If $p <$ alpha risk, reject H_0.

If $p >$ alpha risk, do not reject H_0.

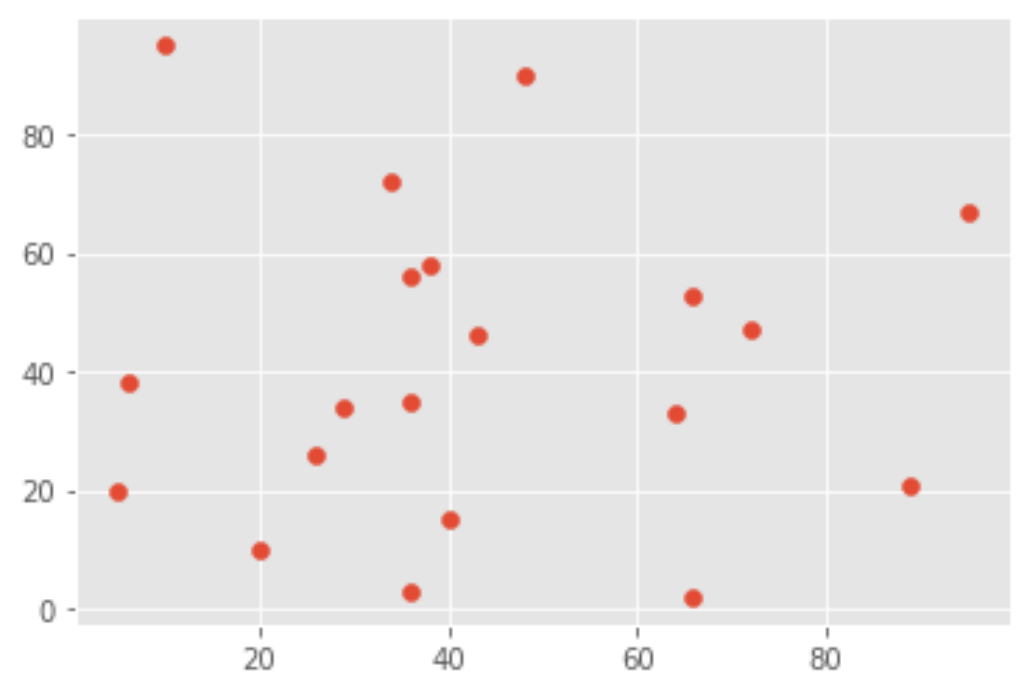

Predictor variable:
Categorical or quantitative?
C
Q
Outcome variable:
Categorical or quantative?
C
Q
Non-parametric test
Comparison of means test
How many groups are being compared?
2
>2
T-test
How many outcome variables?
1
>1
ANOVA
MANOVA
Outcome variable:
Categorical or quantative?
C
Q
Logic regression
How many predictor variables?
1
>1
Simple regression
Multiple regression

Correlation and Regression

X-Y Plots

- The x-y plot or scatter plot represents the pairs of data from two datasets. The horizontal x-axis shows the values from the set x, while the vertical y-axis shows the corresponding values from the set y.

```
>>> import matplotlib.pyplot as plt
>>> plt.style.use('ggplot')
>>> x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
>>> y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]
>>> plt.scatter(x, y)
>>> plt.show()
```



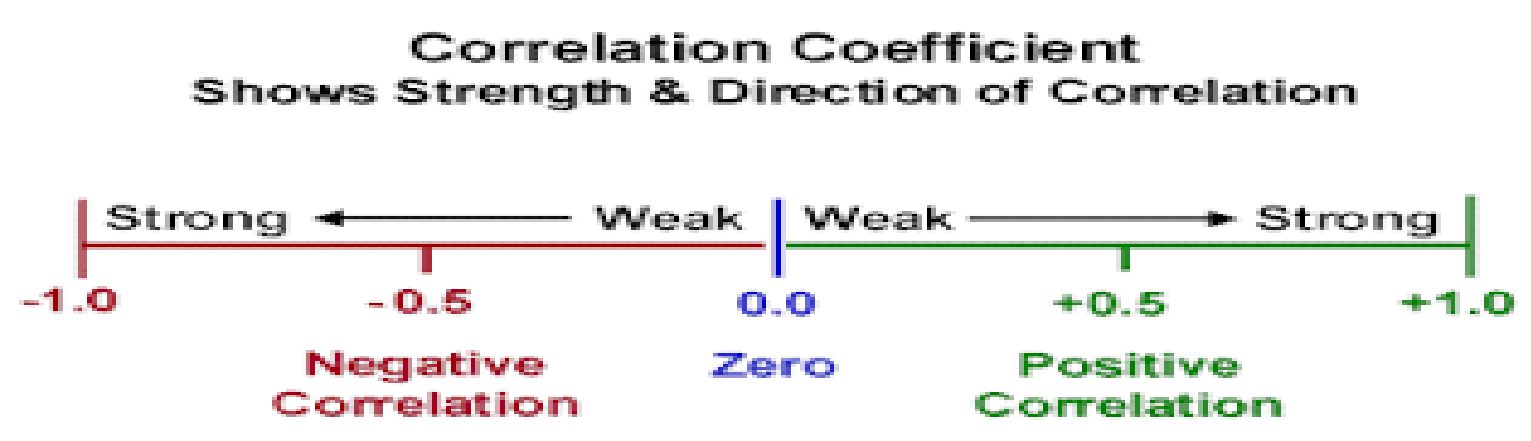
Measures of Correlation

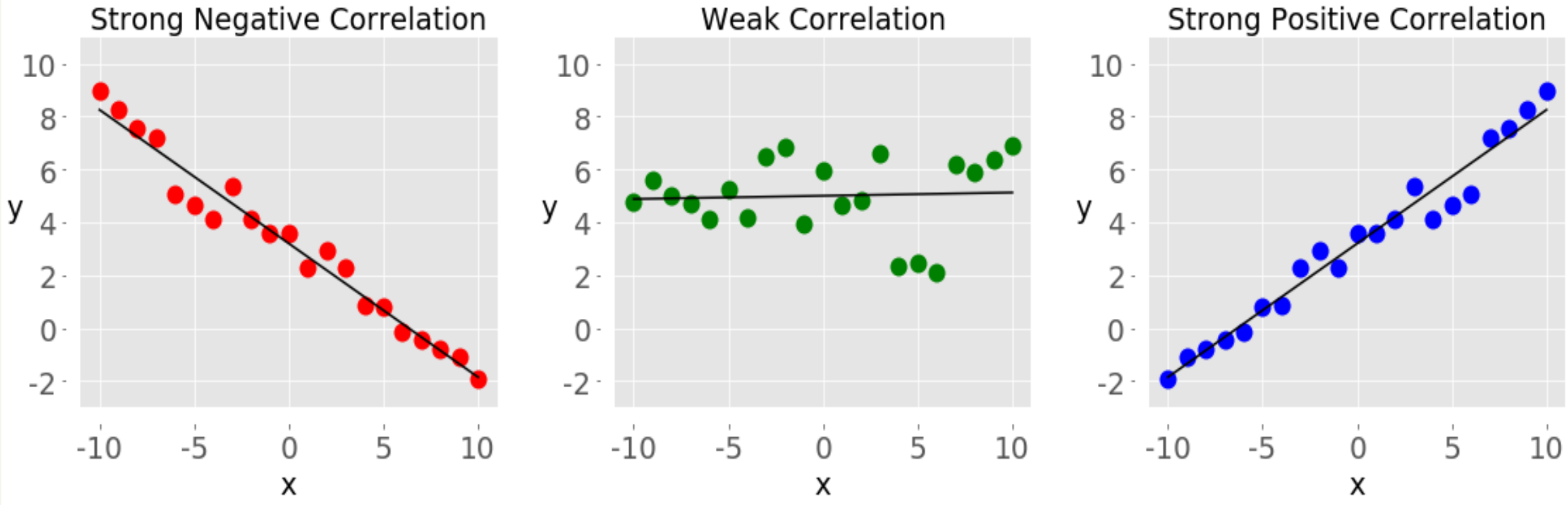
- You'll often need to examine the relationship between the corresponding elements of two variables in a dataset.

- The **correlation coefficient**, or **Pearson product-moment correlation coefficient**, is denoted by the symbol r.

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

- **The value r > 0** indicates positive correlation.
- **The value r < 0** indicates negative correlation.
- **The value r = 1** is the maximum possible value of r. It corresponds to a perfect positive linear relationship between variables.
- **The value r = −1** is the minimum possible value of r. It corresponds to a perfect negative linear relationship between variables.
- **The value r ≈ 0**, or when r is around zero, means that the correlation between variables is weak.




Strong Negative Correlation
Weak Correlation
Strong Positive Correlation
y
x
10
8
6
4
2
0
-2
-10
-5
0
5
10

```
>>> x = list(range(-10, 11))
>>> y = [0, 2, 2, 2, 2, 3, 3, 6, 7, 4, 7, 6, 6, 9, 4, 5, 5, 10, 11, 12, 14]
>>> x_, y_ = np.array(x), np.array(y)

>>> corr_matrix = np.corrcoef(x_, y_)
>>> corr_matrix
array([[1.        , 0.86195001],
       [0.86195001, 1.        ]])
>>> r = corr_matrix[0, 1]
>>> r
0.8619500056316061
```

Testing for the significance of the Pearson correlation coefficient

The Pearson correlation of the sample is r. It is an estimate of rho (ρ), the Pearson correlation of the population. Knowing r and n (the sample size), we can infer whether ρ is significantly different from 0.

- Null hypothesis (H_0): $\rho = 0$
- Alternative hypothesis (H_a): $\rho \neq 0$

To test the hypotheses

- **Step 1:** Calculate the t value (a test statistic)

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$$

- **Step 2: Decide whether to reject the null hypothesis**

❖If p-value < α : The data allows you to reject the null hypothesis.

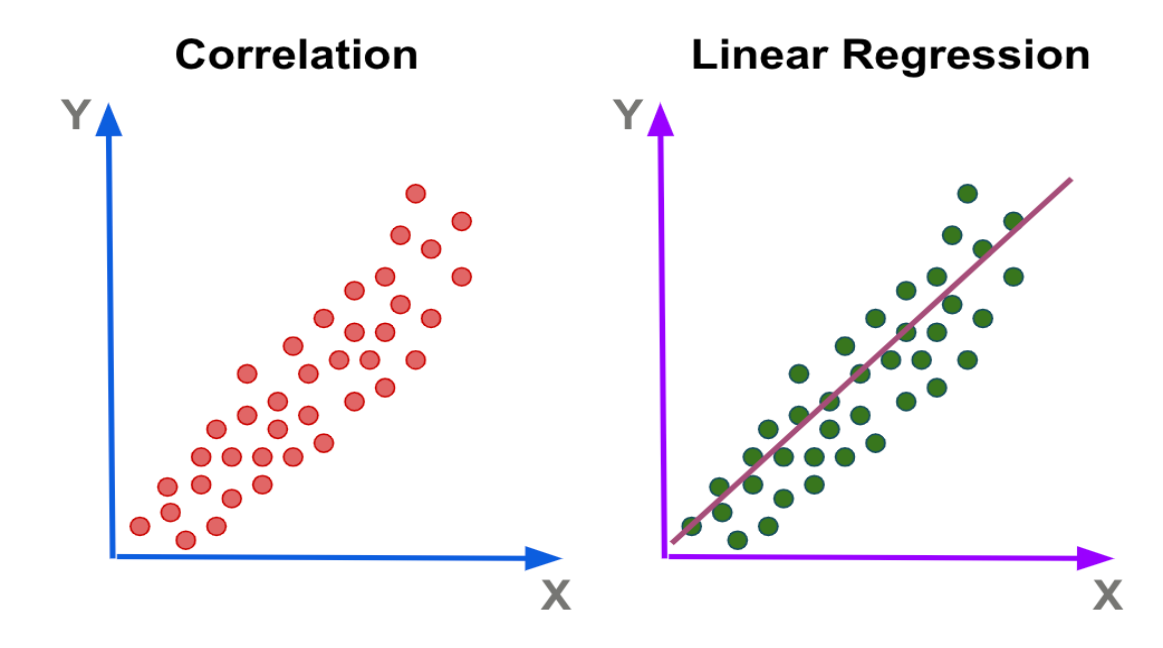
❖If p-value > α : The data doesn't allow you to reject the null hypothesis.

```
>>> scipy.stats.pearsonr(x_, y_)
(0.861950005631606, 5.122760847201171e-07)

>>> scipy.stats.linregress(x_, y_)
LinregressResult(slope=0.5181818181818181,
intercept=5.714285714285714, rvalue=0.861950005631606,
pvalue=5.122760847201164e-07, stderr=0.06992387660074979)
```

Simple linear regression

Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:

- One variable, denoted *x*, is regarded as the **predictor**, **explanatory**, or **independent** variable.
- The other variable, denoted *y*, is regarded as the **response**, **outcome**, or **dependent** variable.



The population regression model:

Dependent Variable

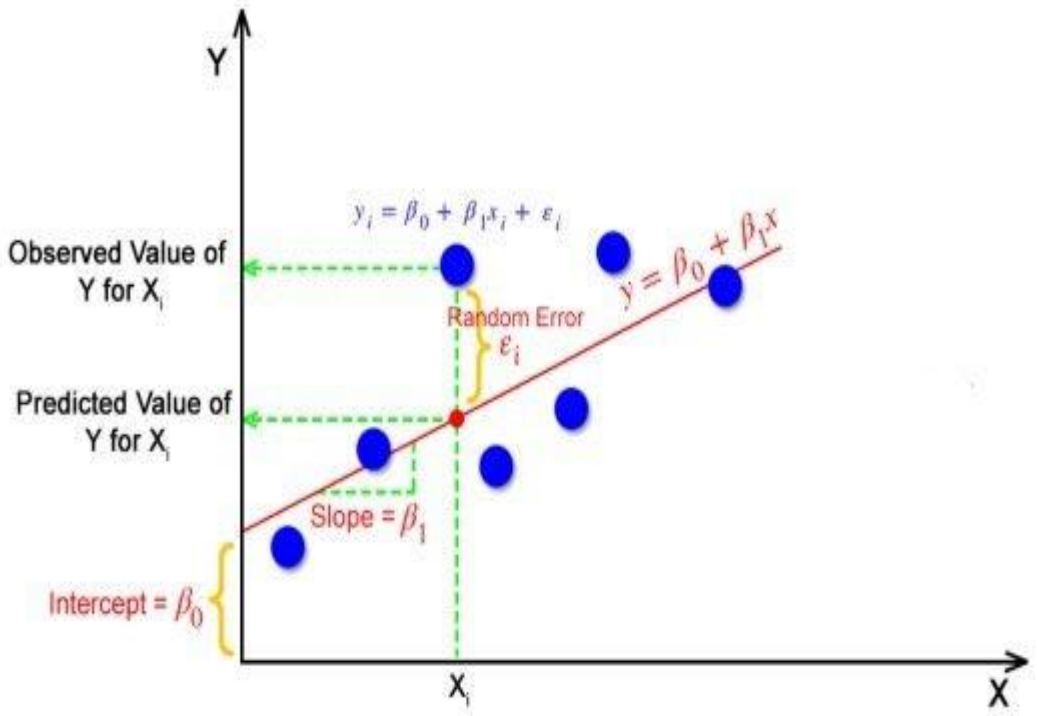
Population y intercept

Population Slope Coefficient

Independent Variable

Random Error term, or residual

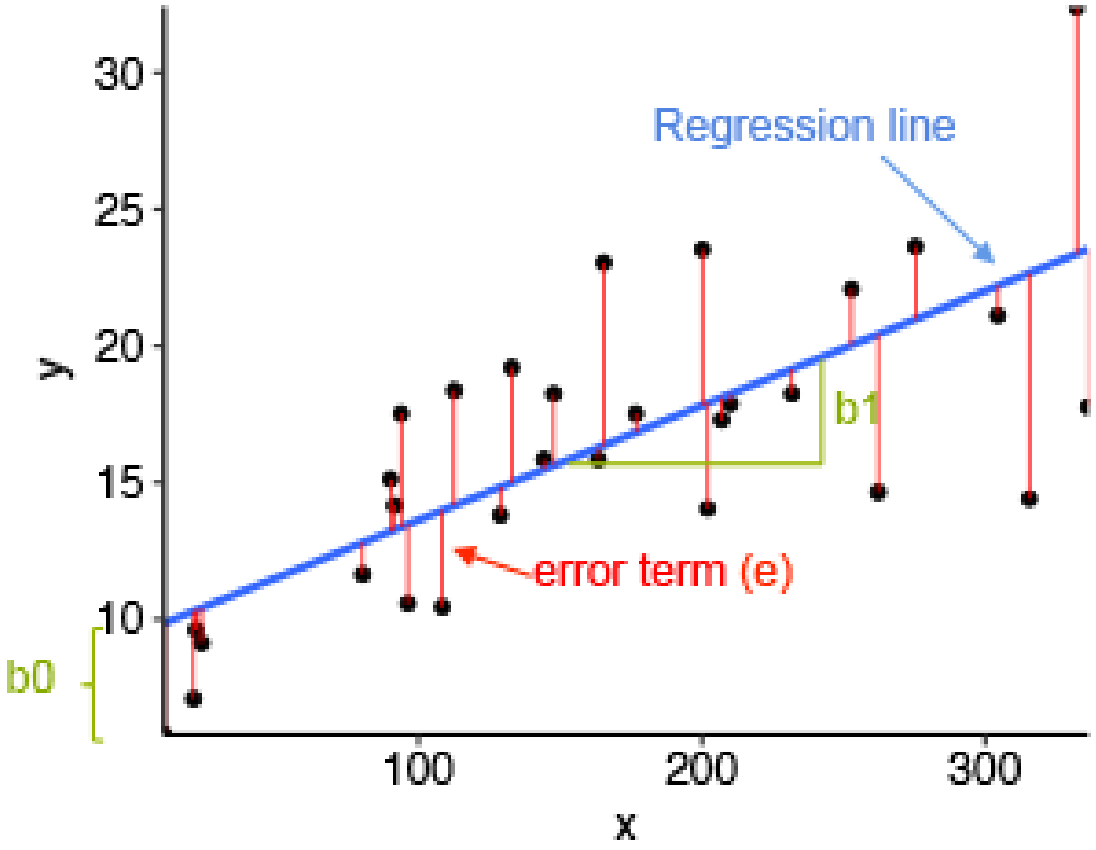
$$Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}; \quad i = 1,2,\ldots,n$$

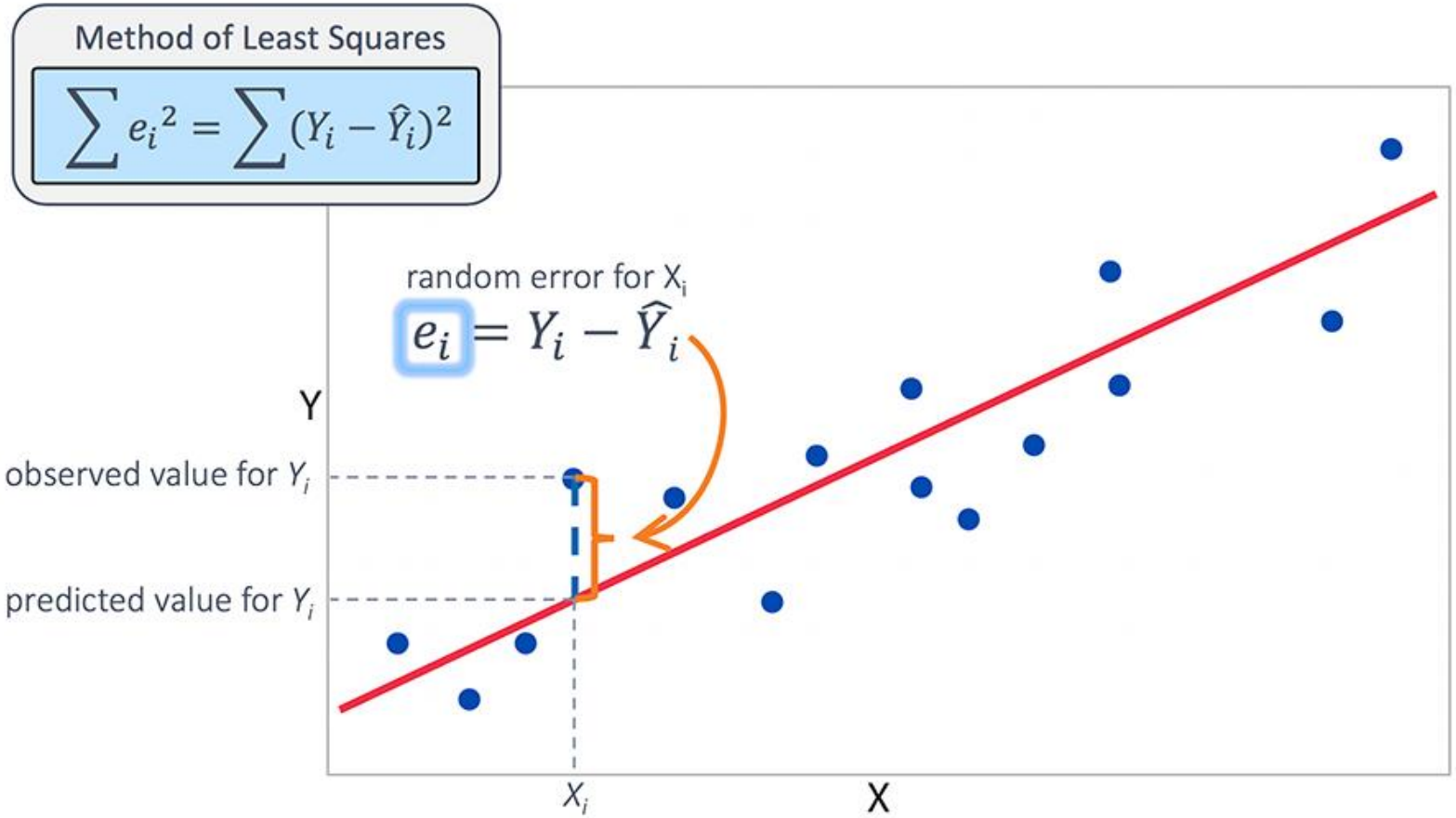


$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$ Population

Regression Coefficients for a . . .

$$\hat{Y}_i = b_0 + b_1 X_i + e_i$$ Sample





$$b_0 = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$b_1 = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

Multiple Linear Regression

- **Multiple linear regression** is used to estimate the relationship between **two or more independent variables** and **one dependent variable**.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = explanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

ANOVA Table for Multiple Linear Regression Model

$$\boldsymbol{H_0: \beta_0 = \beta_1 = \cdots = \beta_p = 0}$$

Source	df	Sum of squares SS	Mean square MS	F	P-value
Model	p	$\sum(\hat{y}_i - \bar{y})^2$ (from data)	MSM=SSM/DFM	MSM/MSE	From Table
Error	$n - p - 1$	$\sum(y_i - \hat{y}_i)^2$ (from data)	MSE=SSE/DFE		
Total	$n - 1$	$\sum(y_i - \bar{y})^2$ (from data)			

T-test: a particular variable is statistically significant in the model

$$H_0: \beta_1 = 0$$
$$H_a: \beta_1 \neq 0$$

TEST STATISTIC

$$t = \frac{b_1}{s_{b_1}}$$

REJECTION RULE

p-value approach:	Reject H_0 if p-value $\leq \alpha$
Critical value approach:	Reject H_0 if $t \leq -t_{\alpha/2}$ or if $t \geq t_{\alpha/2}$

where $t_{\alpha/2}$ is based on a t distribution with $n - 2$ degrees of freedom.

Coefficient of determination

R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, r-squared shows how well the data fit the regression model (the goodness of fit).

$$R^2 = \frac{\sum y_i^2 - \sum ei^2}{\sum y_i^2} = 1 - \frac{\sum e_i^2}{\sum y_i^2}$$

Adjusted R-squared

The adjusted R-squared is a modified version of R-squared that accounts for predictors that are not significant in a regression model. In other words, the adjusted R-squared shows whether adding additional predictors improve a regression model or not.

$$R^2_{Adj.} = 1 - (1 - R^2)\left(\frac{n-1}{n-k-1}\right)$$

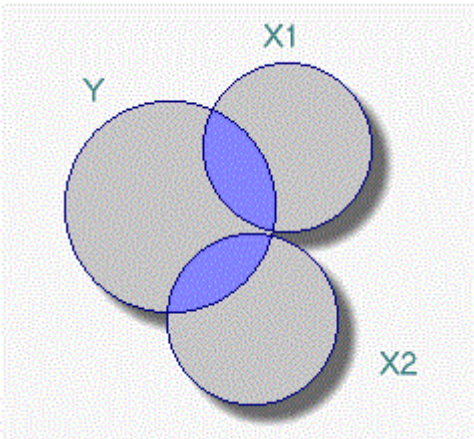
Assumptions of linear regression

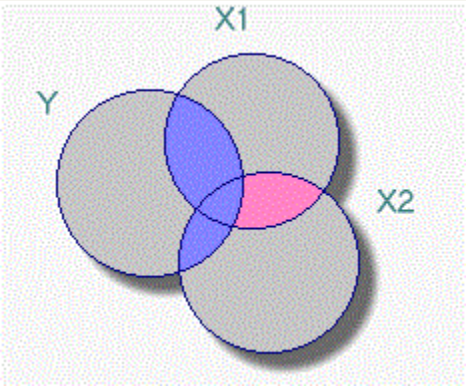
There are four assumptions associated with a linear regression model:

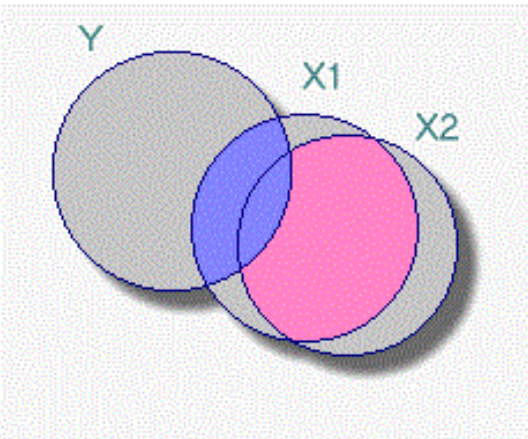
1. **Linearity**: The relationship between X and the mean of Y is linear (x vs y plot).
2. **Homoscedasticity**: The variance of residual is the same for any value of X (residual vs fitted plot).
3. **Independence**: Observations are independent of each other (Durbin-Watson test).
4. **Normality**: For any fixed value of X, Y is normally distributed (Jarque-Bera test).

Collinearity and Multicollinearity

In regression analysis, the collinearity of two variables means that a strong correlation exists between them, making it difficult or impossible to estimate their individual regression coefficients reliably. The extreme case of collinearity, where the variables are **perfectly correlated**, is called **singularity**.







What Is a Variance Inflation Factor (VIF)?

A variance inflation factor (VIF) is a measure of the amount of multicollinearity in regression analysis.

$$\mathrm{VIF}_i = \frac{1}{1 - R_i^2}$$

where:

R_i^2 = Unadjusted coefficient of determination for regressing the ith independent variable on the remaining ones

- VIF equal to 1 = variables are not correlated
- VIF between 1 and 5 = variables are moderately correlated
- VIF greater than 5 = variables are highly correlated

Outlier Detection

- A studentized residual is calculated by dividing the residual by an estimate of its standard deviation.

$$t_i = \frac{\widehat{\varepsilon}_i}{\widehat{\sigma}\sqrt{(n-1)/n}}$$

- An observation with an internally studentized residual that is larger than 3 (in absolute value) is generally deemed an outlier.

Python Packages for Linear Regression

- **NumPy** is a fundamental Python scientific package that allows many high-performance operations on single-dimensional and multidimensional arrays. It also offers many mathematical routines. Of course, it's open-source.
- The package **scikit-learn** is a widely used Python library for machine learning, built on top of NumPy and some other packages. It provides the means for preprocessing data, reducing dimensionality, implementing regression, classifying, clustering, and more. Like NumPy, scikit-learn is also open-source.
- **statsmodels** is a powerful Python package for the estimation of statistical models, performing tests, and more. It's open-source as well.

Linear Regression With statsmodels

You'll start with the simplest case, which is simple linear regression. There are five basic steps when you're implementing linear regression:

1. Import the **packages** you need
2. **Provide data** to work with and eventually **do appropriate transformations**
3. Create a **regression model** and **fit** it with existing data
4. Check the **results** of model fitting to know whether the model is satisfactory
5. Apply the model for **predictions**

Step 1: Import packages

```
>>> import numpy as np
>>> import statsmodels.api as sm
>>> import matplotlib.pyplot as plt
```

Step 2: Provide data and transform inputs

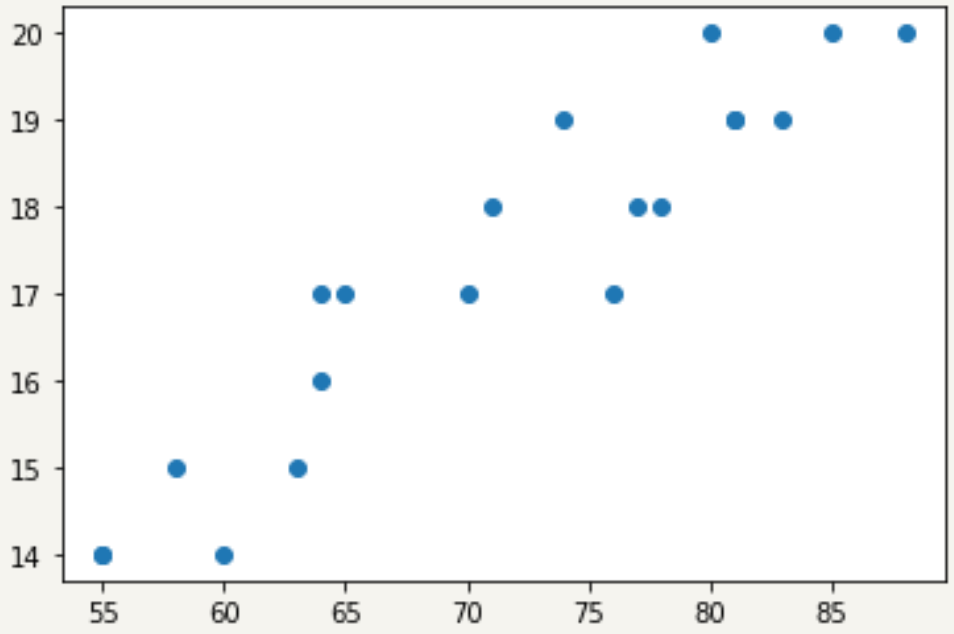
```
x = [ [60,3], [58,3], [80,10], [ 78,9.5], [81,7.5], [65,8], [ 74,8.5], [76,4.5],
[71,3.5], [63,6], [55,4], [64,5], [88,9], [85,7], [64,5], [70,5], [81,6],
[55,4], [83,9], [77,8]]

y =  [14, 15, 20, 18,  19,  17, 19,  17,  18,  15,  14, 17,  20, 20, 16, 17, 19,
14, 19, 18]

>>> x, y = np.array(x), np.array(y)
```

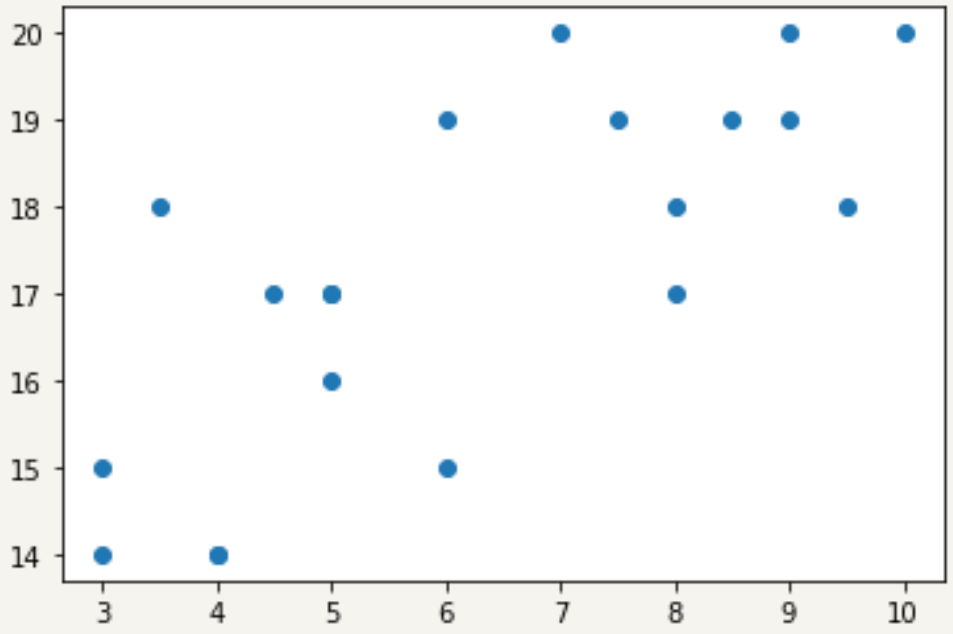
You need to add the column of ones to the inputs if you want statsmodels to calculate the intercept b_0. It doesn't take b_0 into account by default. This is just one function call:

```
>>> x = sm.add_constant(x)
>>> x
array([[ 1. , 60. ,  3. ],
       [ 1. , 58. ,  3. ],
       [ 1. , 80. , 10. ],
       [ 1. , 78. ,  9.5],
       [ 1. , 81. ,  7.5],
       [ 1. , 65. ,  8. ],
       [ 1. , 74. ,  8.5],
       [ 1. , 76. ,  4.5],
       [ 1. , 71. ,  3.5],
       [ 1. , 63. ,  6. ],
       [ 1. , 55. ,  4. ],
       [ 1. , 64. ,  5. ],
       [ 1. , 88. ,  9. ],
       [ 1. , 85. ,  7. ],
       [ 1. , 64. ,  5. ],
       [ 1. , 70. ,  5. ],
       [ 1. , 81. ,  6. ],
       [ 1. , 55. ,  4. ],
       [ 1. , 83. ,  9. ],
       [ 1. , 77. ,  8. ]])
```

```
>>> x1=x[:, 1]
>>> plt.scatter(x1, y)
```

```
>>> x2=x[:, 2]
>>> plt.scatter(x2, y)
```





Step 3: Create a model and fit it

```
>>> model = sm.OLS(y, x)
>>> results = model.fit()
```

Notice that the first argument is the output, followed by the input.

Step 4: Get results

```
>>> print(results.summary())
```

Dep. Variable:	y	**R-squared:**	0.893
Model:	OLS	**Adj. R-squared:**	0.881
Method:	Least Squares	**F-statistic:**	71.08
Date:	Sun, 06 Aug 2023	**Prob (F-statistic):**	5.54e-09
Time:	20:24:18	**Log-Likelihood:**	-19.900
No. Observations:	20	**AIC:**	45.80
Df Residuals:	17	**BIC:**	48.79
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.6810	1.256	3.727	0.002	2.031	7.331
x1	0.1646	0.023	7.204	0.000	0.116	0.213
x2	0.1386	0.103	1.346	0.196	-0.079	0.356

Omnibus:	1.923	**Durbin-Watson:**	2.596
Prob(Omnibus):	0.382	**Jarque-Bera (JB):**	1.121
Skew:	0.231	**Prob(JB):**	0.571
Kurtosis:	1.937	**Cond. No.**	573.

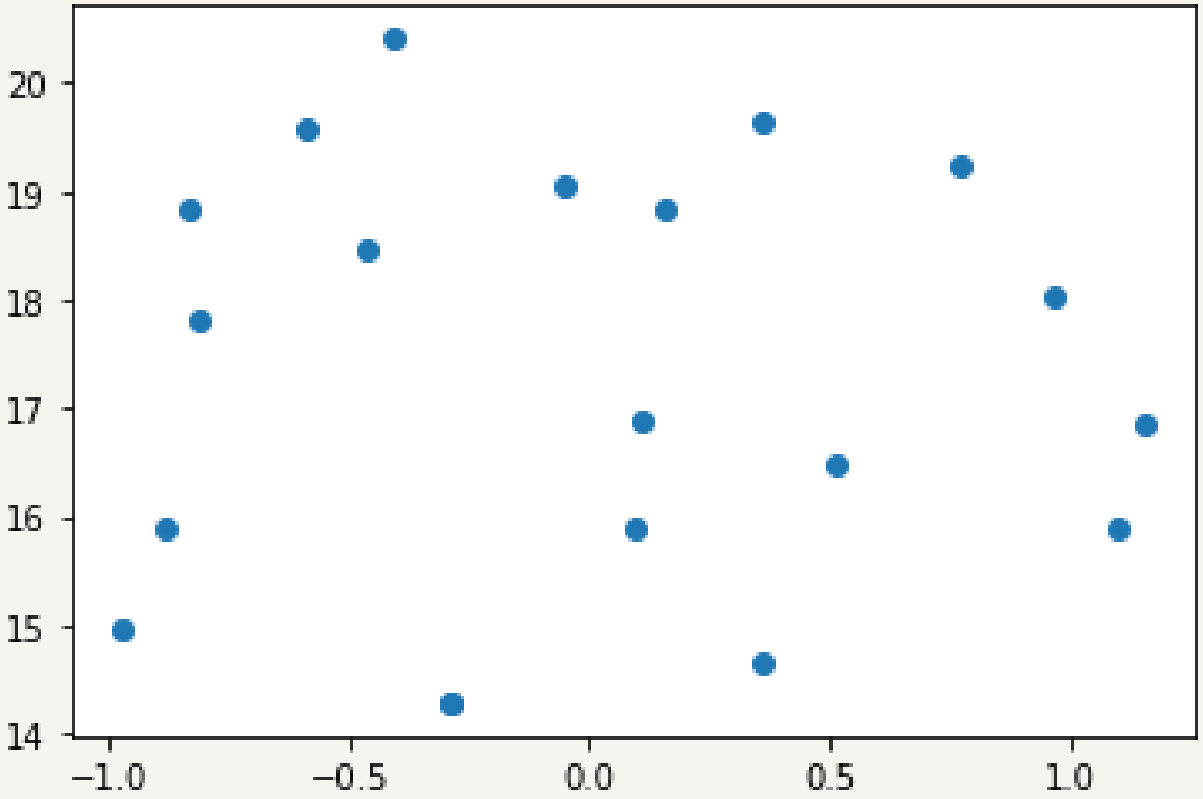
Step 5: Predict response

```
>>> results.fittedvalues
array([14.9702651 , 14.64114619, 19.23131674, 18.83292193, 19.04949671,
       16.48582134, 18.03613232, 17.81104405, 16.84969499, 15.87959884,
       14.28601962, 15.9056065 , 20.40924058, 19.63845862, 15.9056065 ,
       16.89296323, 18.84166901, 14.28601962, 19.58644331, 18.46053479])
>>> results.resid
array([-0.9702651 ,  0.35885381,  0.76868326, -0.83292193, -0.04949671,
        0.51417866,  0.96386768, -0.81104405,  1.15030501, -0.87959884,
       -0.28601962,  1.0943935 , -0.40924058,  0.36154138,  0.0943935 ,
        0.10703677,  0.15833099, -0.28601962, -0.58644331, -0.46053479])
>>> results.predict([1,70,8])
array([17.30861861])
```

```
>>> influence = results.get_influence()
>>> resid_student = influence.resid_studentized_external
>>> resid_student
array([-1.54984625,  0.54166682,  1.23051928, -1.31544876, -0.0712198 ,
       0.81713106,  1.50171753, -1.29743552,  1.96381666, -1.34347637,
      -0.43565561,  1.69025918, -0.62750619,  0.55285335,  0.13436642,
       0.15214082,  0.23719579, -0.43565561, -0.8809071 , -0.66514316])
```

```
>>> from statsmodels.stats.outliers_influence import variance_inflation_factor
>>> variance_inflation_factor(x,1),variance_inflation_factor(x,2)
(2.098540687210218, 2.098540687210218)
```

```
>>> plt.scatter(results.resid, results.fittedvalues)
```

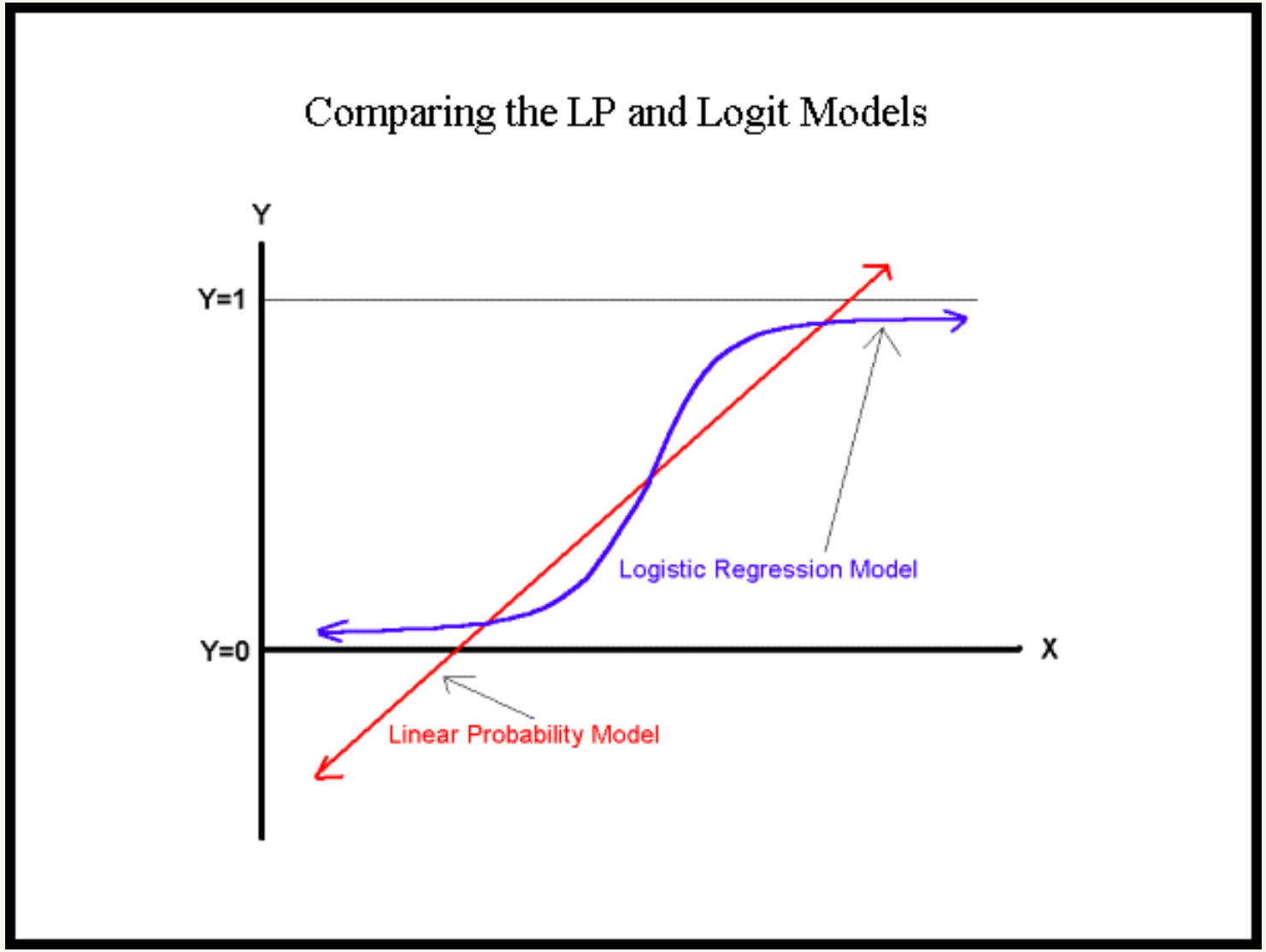


Beyond Linear Regression

- Linear regression is sometimes not appropriate, especially for nonlinear models of high complexity.

- Fortunately, there are other regression techniques suitable for the cases where linear regression doesn't work well. Some of them are support vector machines, decision trees, random forest, and neural networks.

- The package scikit-learn provides the means for using other regression techniques.


Comparing the LP and Logit Models
Y
Y=1
Y=0
X
Logistic Regression Model
Linear Probability Model

Factor Analytics

Factor Analytics is a special technique reducing the huge number of variables into a few numbers of factors is known as factoring of the data, and managing which data is to be present in sheet comes under factor analysis.

- **Types of factor analysis:**

1-Exploratory factor analysis (EFA)

2-Confirmatory factor analysis (CFA)

THANK YOU FOR YOUR ATTENTION

WISH YOU LUCK