

STATISTICS OF COMPUTER SCIENCE

CASE STUDY

EFFECT OF PERFORMANCE ENHANCERS ON SPORTSMEN

Group 10:

LALAM DIVYA SRI	- 20BCS076
MAITREYI	- 20BCS083
SAI TARUN PARASA	- 20BCS096
SHAUNAK MADUSKAR	- 20BCS119
SREEDEVA KRUPANANDA B REDDY	- 20BCS128



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

INDEX

Introduction to dataset

Terminology

Parametric Tests

Implementation of Parametric Tests

Non – Parametric Tests

Implementation of Non-Parametric Tests

Categorical Data Analysis

Implementation of Categorical Data Analysis

Sample Size Estimation

Implementation of Sample Size Estimation

Conclusion

INTRODUCTION TO DATASET

The dataset we took from Kaggle consists of information such as gender, number of sports played and the weights of an athlete before and after taking a particular performance enhancer. Performance enhancers are substances used to improve the performance of a player in certain sports. As per legal matters, such substances are strictly frowned upon. However, they appear to have effects on a person's weight. Thus, the dataset we use covers the effects on the weight of each of these sportsmen.

We chose the aforementioned dataset as it appeared to potential to apply each of the tests we learnt.

The methodology applied begins by estimating the sample size required.

That follows the parametric tests, non-parametric tests and categorical tests.

TERMINOLOGY

Introduction to Parameters and Statistic:

Parameter: This refers to the characteristics of the population. N , μ , p , σ .

Statistic: This refers to the characteristics of the sample drawn from the population. n , \bar{x} , \hat{p} , s

Types of Errors:

Your Statistical Decision	True state of null hypothesis	
	H_0 True (example: the drug doesn't work)	H_0 False (example: the drug works)
Reject H_0 (ex: you conclude that the drug works)	<i>Type I error (α)</i>	<i>Correct</i>
Do not reject H_0 (ex: you conclude that there is insufficient evidence that the drug works)	<i>Correct</i>	<i>Type II Error (β)</i>

Type I error: This takes place when the null hypothesis is true and null hypothesis has been rejected. It is given by a level of significance.

Type II error: This takes place when the null hypothesis is false and the null hypothesis has been accepted. It is given by $(1 - \text{power of the test})$. Power of the test is the probability of rejecting the null hypothesis when the null hypothesis is wrong.

Confidence Intervals:

Confidence Intervals are constructed to be able to estimate a valid acceptable region within which only a certain specified amount of error exists. There are two types of estimations:

- point estimate: this type of estimate gives one value to estimate the actual value.
- Interval estimate: this type of estimate gives a lower limit and an upper limit within which the actual value is expected to lie in.

Advantages of Nonparametric tests:

- Can be used for population parameters that are not normally distributed
- Can be used for nominal or ordinal data
- Can be used to test hypotheses that do not involve population parameters
- Easier computations than the parametric tests
- Easy to understand

Disadvantages of Nonparametric tests:

- Less sensitive than parametric tests
- These tests use less information as compared to parametric tests, thus, leading to lower accuracy
- These tests are less efficient than parametric tests when the distribution is normal.

Chi-square tests:

- Chi-square test for goodness of fit
- Chi-square test for independence
- Chi-square test for homogeneity

Assumptions with Chi-Square test:

- There are no assumptions about the shape of the population distribution
- Chi-square test assumes random sampling.

PARAMETRIC TESTS

Z-tests

- Z-test for one sample mean
- Z-test for two samples mean
- Z-test for one sample proportion
- Z-test for two samples proportion

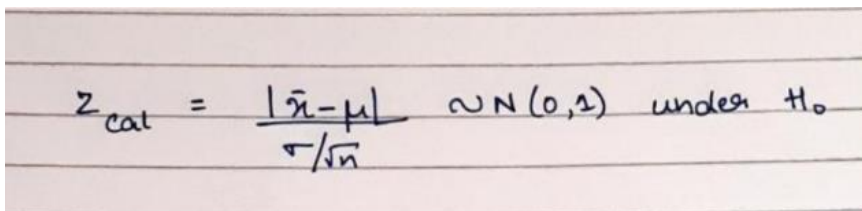
T-tests

- T-test for single sample
- T-test for two independent samples
- T-test for two dependent samples

IMPLEMENTATION OF PARAMETRIC TESTS

- Z-test for one sample, mean

Formula used:



$$Z_{cal} = \frac{|\bar{x} - \mu|}{\sigma / \sqrt{n}} \sim N(0, 1) \text{ under } H_0$$

Code:

z-test for one sample, mean

```
[ ] print('H0: Average number of sports that a person knows is greater than or equal to 8.')
print('H1: Average number of sports that a person knows is not equal to 8.')
alpha = 0.05 #Taking alpha as 5%
sample_size = 50
sample = df['Sports'].sample(sample_size)
sample_mean = np.mean(sample)
sample_SD = np.std(sample)
n = np.shape(sample)[0]
Z_calc = abs(sample_mean - mean)/(sample_SD/(np.sqrt(n-1)))
Z_tab = scipy.stats.norm.ppf(1-alpha)
print(f"Z_calc = {Z_calc} and Z_tab = {Z_tab}")

if Z_calc <= Z_tab:
    print("Null hypothesis is accepted")
else:
    print("Null hypothesis is rejected")

print(f"Confidence interval for {alpha}")
```

```
print(f"Confidence interval for {alpha}")
standard_error = (sample_SD/(np.sqrt(n-1)))*(Z_tab)

lower_limit = sample_mean - standard_error
upper_limit = sample_mean + standard_error
print(f"{lower_limit} <= mean <= {upper_limit}")
```

```
□> H0: Average number of sports that a person knows is greater than or equal to 8.
H1: Average number of sports that a person knows is not equal to 8.
Z_calc = 0.2864109809347407 and Z_tab = 1.6448536269514722
Null hypothesis is accepted
Confidence interval for 0.05
7.538552428271112 <= mean <= 9.261447571728889
```

- Z-test for two samples, mean

Formula used:

$$Z_{cal} = \frac{|\bar{x}_1 - \bar{x}_2|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1) \text{ under } H_0$$

Code:

```
[ ] print('H0 : There is no difference between the means of the populations.')
print('H1 : There is difference between the means of the populations.')
alpha = 0.10
n_1 = 35
n_2 = 40
sample_1 = df['Sports'].sample(n_1)
print(sample_1)
sample_2 = df['Sports'].sample(n_2)
print(sample_2)
sample_1_mean = np.mean(sample_1)
sample_2_mean = np.mean(sample_2)
SD_1 = np.std(sample_1)
SD_2 = np.std(sample_2)
Z_cal = abs(sample_1_mean-sample_2_mean)/np.sqrt((SD_1)**2/n_1 + (SD_2)**2/n_2)
Z_tab = scipy.stats.norm.ppf(1-alpha)
print(f"Z_cal = {Z_cal} and Z_tab = {Z_tab}")
if Z_cal <= Z_tab:
    print("Null hypothesis accepted")
else:
    print("Null hypothesis is rejected")
```

```
[ ] else:
    print("Null hypothesis is rejected")

    print(f"The confidence interval is given by: ")
    margin_error = (np.sqrt((SD_1)**2/n_1 + (SD_2)**2/n_2))*Z_tab
    estimated_diff = (abs(sample_1_mean-sample_2_mean))

    lower_limit = estimated_diff - margin_error
    upper_limit = estimated_diff + margin_error
    print(f"{lower_limit} <= difference_mean <= {upper_limit}")

H0 : There is no difference between the means of the populations.
H1 : There is difference between the means of the populations.
83    5
14    12
93    10
48    8
12    8
80    6
21    6
98    11
95    5
60    5
68    5
26    6
```

```
53    5
45    5
1     6
68    5
38    15
23    6
78    7
54    5
97    8
24    5
88    7
99    5
10    13
12    8
0     6
86    8
84    4
63    9
92    7
Name: Sports, dtype: int64
Z_cal = 0.26293803859022125 and Z_tab = 1.2815515655446004
Null hypothesis accepted
The confidence interval is given by:
-0.7747935843865066 <= difference_mean <= 1.1747935843865052
```

- Z-test for one sample, proportion

Formula used:

$$Z_{cal} = \frac{|\hat{p} - p|}{\sqrt{\hat{p}\hat{q}(1/n)}} \sim N(0,1) \text{ under } H_0$$

Code:

```
[ ] print('H0: 50% of all sportsmen are married')
print('H1: 50% of all sportsmen are not married')
alpha = 0.05 #Taking alpha as 5%
pb=0.50 #estimated proprtion = 50%
sample_size = 80
sample = df['Marital Status'].sample(sample_size)
n = np.shape(sample)[0]
count=0
for x in sample:
    if x=='Married':
        count+=1
p=(count/sample_size)
print("Calculated proportion from sample: ",p)
print((p*(1-p))/n)
Z_calc = abs(p-pb)/np.sqrt((p*(1-p))/n)
Z_tab = scipy.stats.norm.ppf(1-alpha)
print(f"Z_calc = {Z_calc} and Z_tab = {Z_tab}")
```

```
print(f"Z_calc = {Z_calc} and Z_tab = {Z_tab}")

if Z_calc <= Z_tab:
    print("H0 is accepted. There is not enough evidence to reject the claim.")
else:
    print("H0 is rejected. There is enough evidence to reject the claim.")

print(f"Confidence interval for {alpha}")
standard_error = (np.sqrt((p*(1-p))/n))*(Z_tab)

lower_limit = p - standard_error
upper_limit = p + standard_error
print(f"{lower_limit} <= proportion <= {upper_limit}")
```

```
↳ H0: 50% of all sportsmen are married
H1: 50% of all sportsmen are not married
Calculated proportion from sample: 0.475
0.0031171874999999997
Z_calc = 0.44777366283964537 and Z_tab = 1.6448536269514722
H0 is accepted. There is not enough evidence to reject the claim.
Confidence interval for 0.05
0.38316489649478774 <= proportion <= 0.5668351035052122
```

- Z-test for two samples, proportions

Formula used:

$$Z_{cal} = \frac{|\hat{p}_1 - \hat{p}_2|}{\sqrt{\hat{p}\hat{q}\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \sim N(0,1) \text{ under } H_0$$

$$\hat{p} = \frac{x_1 + x_2}{n_1 + n_2}$$

Code:

```
[ ] print('H0: There is no significant difference between the proportions of married male and female sportsmen.')
print('H1: There is a significant difference between the proportions of married male and female sportsmen.')
alpha = 0.05 #Taking alpha as 5%
sample_size = 80
sample = df[['Marital Status','Gender']].sample(sample_size).to_numpy()
Sample1,Sample2=[],[]

for x in sample:
    if x[1]=='M':
        Sample1.append(x[0])
    else:
        Sample2.append(x[0])

def calculateProportion(A):
    count=0
    for x in A:
        if x=='Married':
            count+=1
    return count,len(A)

x1,n1=calculateProportion(Sample1)
```

```
x1,n1=calculateProportion(Sample1)
x2,n2=calculateProportion(Sample2)

p1, p2 = x1/n1, x2/n2

print("Calculated proportions from samples: ")
print(f"Proportion for male sample = {p1} and Proportion for female sample = {p2}")

p=(x1+x2)/(n1+n2)
q=1-p

Z_calc = abs(p1-p2)/np.sqrt(p*q*((1/n1)+(1/n2)))
Z_tab = scipy.stats.norm.ppf(1-alpha)
print(f"Z_calc = {Z_calc} and Z_tab = {Z_tab}")

if Z_calc <= Z_tab:
    print("H0 is accepted. There is not enough evidence to reject the claim.")
else:
    print("H0 is rejected. There is enough evidence to reject the claim.")

print(f"Confidence interval for {alpha}")
```

```
print(f"Confidence interval for {alpha}")
standard_error = (np.sqrt(p*q*((1/n1)+(1/n2))))*(Z_tab)
proportion_difference = p1-p2

lower_limit = proportion_difference - standard_error
upper_limit = proportion_difference + standard_error

print(f"{lower_limit} <= difference in proportions <= {upper_limit}")
```

```
□> H0: There is no significant difference between the proportions of married male and female sportsmen.
H1: There is a significant difference between the proportions of married male and female sportsmen.
Calculated proportions from samples:
Proportion for male sample = 0.4878048780487805 and Proportion for female sample = 0.5641025641025641
Z_calc = 0.6830683280463458 and Z_tab = 1.6448536269514722
H0 is accepted. There is not enough evidence to reject the claim.
Confidence interval for 0.05
0.38316489649478774 <= difference in proportions <= 0.10742994481516766
```

- T-test for single sample

Formula used:

$$t_{cal} = \frac{|\bar{x} - \mu|}{s/\sqrt{n}} \sim t(n-1) \text{ d.f under } H_0$$

Code:

```
[ ] print('H0: Average weight of a person before taking enhancers is equal to 70.')
    print('H1: Average weight of a person before taking enhancers is not equal to 70.')
    sample_size = 15
    population = df['Weight before'].to_numpy()
    mean = population.mean()
    t_sample = df['Weight before'].sample(sample_size)
    alpha = 0.05
    n = np.shape(t_sample)[0]
    dof = n-1
    sample_mean = t_sample.mean()
    sample_SD = t_sample.std()
    t_tab = scipy.stats.t.ppf(q=1-alpha,df=dof)
    t_cal = abs(sample_mean - mean)/(sample_SD/(np.sqrt(n-1)))

    print(f"t_cal = {t_cal} and t_tab = {t_tab}")
    if t_cal <= t_tab:
        print("Accept Null hypothesis")
    else:
        print("Null hypothesis rejected")

H0: Average weight of a person before taking enhancers is equal to 70.
H1: Average weight of a person before taking enhancers is not equal to 70.
t_cal = 0.15781401445339294 and t_tab = 1.7613101357748562
Accept Null hypothesis
```

- T-test for two independent samples

Formula Used:

$$t_{cal} = \frac{\bar{D}}{s_D/\sqrt{n}} \sim t(n-1) \text{ d.f under } H_0$$

Code:

```

▶ print('H0 : Both samples are from the same population.')
print('H1 : Both samples are from different populations.')
alpha = 0.05
n_1 = 11
n_2 = 16
dof = n_1 + n_2 - 2
sample_1 = df['Weight before'].sample(n_1)
sample_2 = df['Weight before'].sample(n_2)
sample_1_mean = np.mean(sample_1)
sample_2_mean = np.mean(sample_2)
SD1 = sample_1.std()
SD2 = sample_2.std()
t_tab = scipy.stats.t.ppf(q=1-alpha,df=dof)
t_cal = (abs(sample_1_mean-sample_2_mean))/(np.sqrt((SD1)**2/n_1 + (SD2)**2/n_2))
print(f"t_cal = {t_cal} and t_tab = {t_tab}")
if t_cal <= t_tab:
    print("Accept Null hypothesis")
else:
    print("Null hypothesis rejected")

```

```

else:
    print("Null hypothesis rejected")

☐ H0 : Both samples are from the same population.
H1 : Both samples are from different populations.
t_cal = 0.11238987737236113 and t_tab = 1.7081407612518986
Accept Null hypothesis

```

- T-test for two dependent samples

Formula Used:

$$t_{cal} = \frac{|\bar{x}_1 - \bar{x}_2|}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{(n_1+n_2-2)} \quad \text{df under } H_0$$

$$S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{(n_1+n_2-2)}}$$

Code:

```
▶ print('H0 : Weight of a person after the test is the same as the one before.')
print('H1 : Weight of a person after the test is different from the one before')
n = 12
alpha = 0.05
dof = n-1
sample = df[['Weight before', 'Weight after']].sample(n)
data = sample['Weight before'] - sample['Weight after']
diff = data.to_numpy()
sample_mean = diff.mean()
sample_SD = diff.std()
t_tab = scipy.stats.t.ppf(q=1-alpha, df=dof)
t_cal = abs(sample_mean - 0)/(sample_SD/(np.sqrt(n)))
print(f"t_cal = {t_cal} and t_tab = {t_tab}")
if t_cal <= t_tab:
    print("Accept Null hypothesis")
else:
    print("Null hypothesis rejected")
```



```
↳ H0 : Weight of a person after the test is the same as the one before.
H1 : Weight of a person after the test is different from the one before
t_cal = 2.5041900371049084 and t_tab = 1.7958848187036691
Null hypothesis rejected
```

IMPLEMENTATION OF NON-PARAMETRIC TESTS

- Runs test

Code:

```

print('H0 : The data is random')    ### Claim
print('H1 : The data is not random')

def runsTest(l, l_median) :          ### Function For Runs Test

    runs, n1, n2 = 0, 0, 0           ### Initially all variables initialised with 0

    # Checking for start of new run
    for i in range(len(l)):

        # no. of runs
        if (l[i] >= l_median and l[i-1] < l_median) or (l[i] < l_median and l[i-1] >= l_median):
            runs += 1

        # no. of positive values
        if(l[i]) >= l_median:
            n1 += 1

        # no. of negative values
        else:
            n2 += 1

```

```

        # no. of negative values
        else:
            n2 += 1

    runs_exp = ((2*n1*n2)/(n1+n2))+1
    st_dev = math.sqrt((2*n1*n2*(2*n1*n2-n1-n2)) / (((n1+n2)**2)*(n1+n2-1)))

    z = (runs-runs_exp)/st_dev

    return z

n = 50
data = df['Weight after'].sample(n).to_numpy()

l_median = statistics.median(data)

Z_cal = abs(runsTest(data, l_median))
print('\ntest-statistic =', Z_cal)

alpha = 0.05
Z_tab = scipy.stats.norm.ppf(1-alpha)

if abs(Z_cal) > Z_tab :
    print("\nNull hypothesis is rejected.")
else:

```

```

data = arr[weight_after].sample(n).to_numpy()

l_median = statistics.median(data)

Z_cal = abs(runsTest(data, l_median))
print('\ntest-statistic =', Z_cal)

alpha = 0.05
Z_tab = scipy.stats.norm.ppf(1-alpha)

if abs(Z_cal) > Z_tab :
    print("\nNull hypothesis is rejected.")
else:
    print("\nNull hypothesis is accepted.")

```



H0 : The data is random
H1 : The data is not random

Z-statistic = 1.0557360116906864

Null hypothesis is accepted.

- Sign test for one sample

Formula used:

$$Z = \frac{(X + 0.05) - (n/2)}{\sqrt{n/2}}$$

Code:

```

[ ] print('H0 : Median is equal to the hypothesized value') ### Claim
    print('H1 : Median is not equal to the hypothesized value')

def oneSampleSignTest(data, claim) :                                ### Function to Calculate One Sample Sign test
    sample_size = len(data)
    plus_count, minus_count = 0, 0

    for i in data:                                                  ### Loop helps to check the data
        if i > claim:
            plus_count += 1
        elif i < claim:
            minus_count += 1

    if(sample_size < 26):
        test_value = min(plus_count, minus_count)
        return [test_value, plus_count, minus_count]
    else:
        X = min(plus_count, minus_count)
        z_calculated = ((X + 0.05) - (sample_size/2))/np.sqrt(sample_size/2)
        return [z_calculated, plus_count, minus_count]

```

```
[ ] def showResults(test_value, plus_count, minus_count, sample_size):
    n = plus_count + minus_count
    print(f"\nThe total number of plus and minus signs omit the zeros: {n}")
    print(f"Test value: {test_value}")

    if(sample_size < 26):

        # At 95% Confidence level and 2-tailed test the critical value is 4.

        if test_value <= 4:
            print("\nNull hypothesis is rejected.")
        else:
            print("\nNull hypothesis is accepted.")
    else:

        # At 95% Confidence level and 2-tailed test the critical value is 1.96.

        if abs(test_value) >= 1.96 :
            print("\nNull hypothesis is rejected.")
        else:
            print("\nNull hypothesis is accepted.")

n = 27
```

```
n = 27
data = df['Weight before'].sample(n).to_numpy()

[test_value, plus_count, minus_count] = oneSampleSignTest(data, 11)
showResults(test_value, plus_count, minus_count, len(data))
```

● H0 : Median is equal to the hypothesized value
H1 : Median is not equal to the hypothesized value

The total number of plus and minus signs omit the zeros: 27
Test value: -3.6606263378259714

Null hypothesis is rejected.

- Wilcoxon Rank Sum test

Formula Used:

$$Z = \frac{R - \mu_R}{\sigma_R}$$

$$\mu_R = \frac{n_1(n_1 + n_2 + 1)}{2}$$

$$\sigma_R = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$$

Code:

```
[ ] print('H0 : There is a difference in weights for males and females')    ### Claim
    print('H1 : There is no difference in weights for males and females')

    from scipy import stats
    alpha = 0.05
    z_tab=-1.96

    x = df['Weight before'].to_numpy()
    y = df['Weight after'].to_numpy()

    results = stats.ranksums(x, y)
    z = results[0]
    print("\nz_calc = ",z)

    if( z < z_tab):
        print("\nNull hypothesis is Rejected.")
    else:
        print("\nNull hypothesis is Accepted.")
```

```
H0 : There is no difference in weights for males and females
H1 : There is a difference in weights for males and females
```

```
if( z < z_tab):
    print("\nNull hypothesis is Rejected.")
else:
    print("\nNull hypothesis is Accepted.")
```

```
• H0 : There is no difference in weights for males and females
  H1 : There is a difference in weights for males and females
```

```
z_calc = -0.34573952752834375
```

```
Null hypothesis is Accepted.
```

- Wilcoxon Sign Rank test

Formula Used:

$$Z = \frac{W_S - \mu_R}{\sigma_R}$$

$$\mu_R = \frac{n(n+1)}{4}$$

$$\sigma_R = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

Code:

```
[ ] print("H0: The sample selected is random.")      ### Claim
    print("H1: The sample selected is not random.")

    from scipy import stats
    alpha = 0.05
    z_tab=-1.96

    x = df['Weight before'].to_numpy()
    y = df['Weight after'].to_numpy()

    results = stats.wilcoxon(x, y)
    z = results[0]
    print("\nz_calc = ",z)

    if( z < z_tab):
        print("\nNull hypothesis is Rejected.")
    else:
        print("\nNull hypothesis is Accepted.")
```



```
H0: The sample selected is random.
H1: The sample selected is not random.
```

```
z_calc = 2409.0
```

```
Null hypothesis is Accepted.
```

IMPLEMENTATION OF CATEGORICAL DATA ANALYSIS

- Chi-square test for goodness of fit

Formula Used:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Code:

```
from scipy.stats import chi2
def chi_square_test(obs, exp, alpha=0.05):
    n = len(obs)
    chi_square = 0
    for i in range(n):
        chi_square += (((obs[i] - exp[i])**2)/exp[i])

    cv = chi2.ppf(1-alpha, n-1)
    newline = '\n'
    print(f"alpha= {alpha}{newline}df= {n-1}{newline}Chi-Square= {round(chi_square, 4)}{newline}Critical Value= {round(cv, 3)}{newline}")
    if(chi_square > cv):
        return "The decision is to reject the null hypothesis"
    else:
        return "The decision is to accept the null hypothesis"
```

A researcher wished to see the number of sports/games played by a person in a street. A sample of 100 people provided the data.

```
[ ] count1, count2, count3, count4 = 0, 0, 0, 0
for i in sports:
    if(i <= 6):
        count1 += 1
```

```
[ ] count1, count2, count3, count4 = 0, 0, 0, 0
for i in sports:
    if(i <= 6):
        count1 += 1
    elif(i < 9 and i > 6):
        count2 += 1
    elif(i >= 9 and i < 14):
        count3 += 1
    else:
        count4 += 1
```

```
obs = [count1, count2, count3, count4]
exp = [25, 25, 25, 25]
print("Observed: ", obs)
print("Expected: ", exp)
```

```
Observed: [38, 26, 27, 9]
Expected: [25, 25, 25, 25]
```

```
[ ] print("H0: There is no preference in number of sports played")
print("Ha: There is preference in number of sports played")
chi_square_test(obs, exp, 0.05)
```

```
[ ] print("H0: There is no preference in number of sports played")
    print("Ha: There is preference in number of sports played")
    chi_square_test(obs, exp, 0.05)
```

```
H0: There is no preference in number of sports played
Ha: There is preference in number of sports played
alpha= 0.05
df= 3
Chi-Square= 17.2
Critical Value= 7.815
```

```
'The decision is to reject the null hypothesis'
```

- Chi-square test for independence

Formula Used:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Code:

The proportion in change of weight of each sportsmen is same for male and female.

Groups	x<7	7<=x<11	x>=12
Yes	12	23	56
No	88	77	44

```
[ ] print("H0: The proportion in change of weight of each sportsmen is equal")
    print("Ha: The proportion in change of weight of each sportsmen is not equal")
    matrix = [[12, 23, 56], [88, 77, 44]]
    test_homogeneity(matrix)
```

```
H0: The proportion in change of weight of each sportsmen is equal
Ha: The proportion in change of weight of each sportsmen is not equal
Alpha= 0.05
Degrees of Freedom= 2
Chi_square= 49.624
Critical Value= 3.841
```

```
'The decision is to reject the null hypothesis'
```

- Chi-square test for homogeneity

Formula Used:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Code:

A researcher wants to check if there is any relationship between weight and gender.

H0: Weight and gender of an individual is independent. Ha: Weight and gender of an individual is dependent.

Groups	Thin	Normal	Fat	Total
Male	9	30	11	50
Female	10	29	11	50

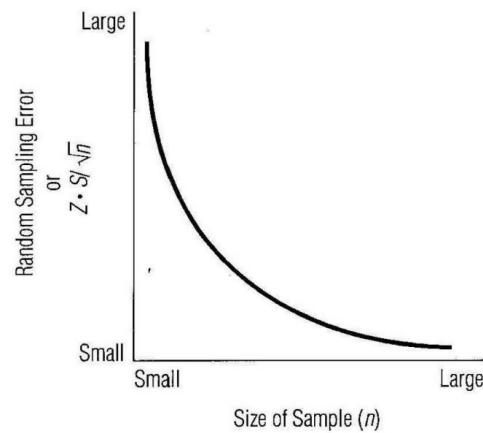
```
[ ] matrix = [obs, exp]
print("H0: Weight of an individual is independent of gender of individual")
print("Ha: Weight of an individual is dependent of gender of individual")
print(test_independence(matrix))
```

```
H0: Weight of an individual is independent of gender of individual
Ha: Weight of an individual is dependent of gender of individual
Alpha= 0.05
Degrees of Freedom= 2
Chi_square= 0.07
Critical Value= 3.841
```

The decision is to accept the null hypothesis

SAMPLE SIZE ESTIMATION

Relationship between Sample Size and Error



IMPLEMENTATION OF SAMPLE SIZE ESTIMATION

- Single Sample using mean

Formula Used:

$$n = \left(\frac{zs}{d} \right)^2$$

Code:

```
def single_mean(alpha):
    z_tab = scipy.stats.norm.ppf(1-alpha)
    s = 2.5
    d = 0.3
    n = ((z_tab * s) / d) ** 2
    print(f"\n> Sample size for one sample single mean is {round(n)}.")
```

```
[130] single_mean(0.05)
```

```
> Sample size for one sample single mean is 188.
```

- Two Samples using mean

Formula Used:

$$N = n_1 + n_2 = \frac{4\sigma^2(z_{1-\alpha/2} + z_{1-\beta})^2}{(d = \mu_1 - \mu_2)^2}$$

Code:

```
def single_proportion(alpha):
    z_tab = scipy.stats.norm.ppf(1-alpha)
    p = 0.70
    d = 0.25
    n = ((z_tab** 2) * p * (1 - p)) / (d ** 2)
    print(f"\n> Sample size for one sample single proportion is {round(n)}.")
```

```
[131] two_means(0.10, '0.80')
```

```
> Sample size for two sample two means is 690 per group.
```

- Single Sample using proportion

Formula Used:

$$n = \frac{Z^2 pq}{E^2}$$

Code:

```
def single_proportion(alpha):
    z_tab = scipy.stats.norm.ppf(1-alpha)
    p = 0.70
    d = 0.25
    n = ((z_tab** 2) * p * (1 - p)) / (d ** 2)
    print(f"\n> Sample size for one sample single proportion is {round(n)}.")
```

```
[1] single_proportion(0.01)
```



```
> Sample size for one sample single proportion is 18.
```

- Two Samples using proportion

Formula Used:

$$N = n_1 + n_2 = \frac{4(z_{1-\alpha/2} + z_{1-\beta})^2 \left[\left(\frac{P_1 + P_2}{2} \right) \left(1 - \frac{P_1 + P_2}{2} \right) \right]}{(d = P_1 - P_2)^2}$$

Code:

```
def two_proportions(alpha, beta):
    za = scipy.stats.norm.ppf(1-alpha)
    zb = power[beta]
    p1 = 0.65
    p2 = 0.72
    P = (p1 + p2) / 2
    d = p1 - p2
    N = 4 * (za + zb) ** 2 * (P * (1 - P)) / (d ** 2)
    print(f"\n> Sample size for two sample two proportions is {round(round(N) / 2)} per group.")
```

```
[133] two_proportions(0.10, '0.90')
```

```
> Sample size for two sample two proportions is 579 per group.
```

CONCLUSION

After performing all the aforementioned tests on the dataset, certain inferences were made pertaining to the dataset. We could decisively prove the assumptions we made on certain aspects of the dataset we chose to study. The results we obtained are as follows

1. The average number of sports played by the respondents was at least 8
2. The proportions of sportsmen on the basis of marital status were equal
3. The proportions of married sportsmen by gender were equal
4. The average weight of a person before taking the enhancer was about 70 kg
5. The performance enhancers have no side effects on the weight of the person
6. The data sampled from the dataset is completely random in nature
7. The weight is equally distributed between males and females
8. The weight loss is not correlated with the number of sports played