Sample t test / Sample Z test

1. one-sample t test / one-sample Z test
2. two-sample t test / two-sample Z test

- uses ztest
- used to compare mean against a claimed value

one-sample t test / one-sample Z test

- one-sample t test is used to comapre the mean/average of a sample against a claimed value and establish whether the difference between them is statistically significant or not
- z-test
    - population S.D. should be known
    - we work with sample S.D.
    - Sample size, n > 30
- t-test
    - Sample S.D should be known
    - Sample size, n <= 30
- data should follow normal distribution

import statsmodels.stats.weightstats

```
import numpy as np
import pandas as pd
# import numpy & pandas

import statsmodels
from statsmodels import stats
from statsmodels.stats import weightstats
# for ztest
```

np.mean(ndarray)

```
x = [3, 7, 6, 5, 9, 8, 2, 3, 5, 4, 6, 7]
# create a list of Sample data
```

- Q1.
- I'm claiming that the mean of x is more than 6.5
- H_o : mean(x) >= 6.5
- H_A / `H1` : mean(x) < 6.5

```
np.mean(x)
# actual sample Mean x̄ to be used to compare with claimed mean
# and then cestablish whether difference is staistically significant or not
```

```
5.416666666666667
```

statsmodels.stats.weightstats.ztest(Sample, value=ClaimValue, alternative='Sign of Hᴀ')

```
statsmodels.stats.weightstats.ztest(x, value=6.5, alternative='smaller')
# statsmodels.stats.weightstats.ztest(list/ndarray, value=Claimed Value, alternative=Sign Of Hᴀ)
# returns (TestStatistics, P-Value)
```

```
(-1.744291601622091, 0.04055412767721645)
```

- assume `C.L. = 98%` means `LoS = 2% = 0.02`
- `P-value = 0.04055` > `LoS = 0.02`
- since `P-Value` > `LoS`, we don't reject this H_o

- `P-value = 0.04055` means there is 4.055% chance that we commit Type 1 error
- means 4.055% chances of chance variation so that we might reject this hypothesis

- if `p-value < 0.05 = 5%` , we reject the H_o ,
- it means that the difference between Sample Statistics `SS` & Population Parameter `PP` is large enough not to be ignored, so action is required

Q2.

- Now, I'm claiming, that mean of x is more than 6.0
- H_o : mean(x) >= 6.0

- `Hᴀ` / `H1` : mean(x) < 6.0

```
np.mean(x)
# actual sample Mean x̄ to be used to compare with claimed mean
# and then cestablish whether difference is stistically significant or not
```

```
5.416666666666667
```

```
statsmodels.stats.weightstats.ztest(x, value=6.0, alternative='smaller')
# statsmodels.stats.weightstats.ztest(list/ndarray, value=Claimed Value, alternative=Sign Of Hᴀ)
# returns (TestStatistics, P-Value)
```

```
(-0.9392339393349719, 0.17380532364640544)
```

- assume `C.L. = 98%` means `LoS = 2% = 0.02`
- `P-value = 0.1738` > `LoS = 0.02`
- since `P-Value` > `Los` , we don't reject this `Hₒ`
- but, `p-Value = 0.1735 or 17.35%` , which is very high (although `> 5% LoS` is acceptable), so we reject this Null Hypothesis

- if `p-value < 0.05 = 5%` , we reject the `Hₒ` ,
- it means that the difference between Sample Statistics `SS` & Population Parameter `PP` is large enough not to be ignored, so action is required

using Sample S.D. to manually calculate Test Statistics

np.mean(ndarray)

- actual Population Mean (μ)

```
np.mean(x)
# actual Population Mean (μ)
# to be used to compare with claimed mean
# and then cestablish whether difference is stistically significant or not
```

```
5.416666666666667
```

np.std(ndarray, ddof=1)

- Sample Standard Deviation (σ)

```
np.std(x, ddof=1)
# Sample Standard Deviation (σ)
```

```
2.151461800448216
```

len(ndaray)

- Calculate Sample Size (n)

```
len(x)
# Claculate Sample Size (n)
```

```
12
```

Test Statistics formula

- `TestStatistics, Z = (PopulationMean - ClaimedMean) / S.D. / sqrt(SampleSize)`
- `TestStatistics, Z = (μ - X) / ( σ/sqrt(n) )`
- `TestStatistics, Z = (μ - X) / ( σ/(n)**0.5 )`

```
(5.416666 - 6.5) / (2.151461800448216/12**0.5)
# (μ - X) / (S.D./n**0.5)
# test statistics if claimValue=6.5
```

```
-1.7442926750323076
```

```
(5.416666 - 6) / (2.151461800448216/12**0.5)
# (μ - X) / (S.D./n**0.5)
# test statistics if claimValue=6.0
```

```
-0.9392350127451884
```

import scipy.stats.norm

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

norm.cdf(Z)

```
norm.cdf(-1.7442926750323076)
# to find out area under curve using value on x-axis
# returns P-Value when claimValue=6.5
```

```
0.040554034138080716
```

```
norm.cdf(-0.9392350127451884)
# to find out area under curve using value on x-axis
# returns P-Value when claimValue=6.0
```

```
0.17380504814940417
```

95% upper bound

- `5.41` is the Population mean and because error is on left side, we would prefer this to be as high as possible
- `95% upper bound` : the highest possible value at 95% C.L., is called as 95% upper bound

norm.ppf(area/prob)

```
norm.ppf(0.05)
# value on x-axis fo area under curve for 0.05 area
# value on x-axis is on left side of mean so it returns -ve value
```

```
-1.6448536269514729
```

- 95% upper bound = `PopulationMean + (TestStatistics * S.D / Sqrt(SampleSize) )`
- 95% upper bound = `μ + (Z * σ / Sqrt(n) )`
- 95% upper bound = `μ + (Z * σ / n**0.5 )`

```
5.41 + 1.645*(2.151461800448216 / 12**0.5)
# 95% upper bound = Population Mean + TestStatistics * S.D / Sqrt(SampleSize)
# 95% upper bound = μ + Z * σ / Sqrt(n)
# 95% upper bound = μ + Z * σ / n**0.5

# if considering chance variation or benefit of doubt,
# actual mean can go till 6.431665948328879
# but it cannot be exactly 6.431665948328879
# which is answer for highest possible sample mean or 95% upper bound
```

```
6.431665948328879
```

```
statsmodels.stats.weightstats.ztest(x, value=6.0, alternative='smaller')
```

```
(-0.9392339393349719, 0.17380532364640544)
```

```
statsmodels.stats.weightstats.ztest(x, value=6.5, alternative='smaller')
```

```
(-1.744291601622091, 0.04055412767721645)
```

```
statsmodels.stats.weightstats.ztest(x, value=7.0, alternative='smaller')
```

```
(-2.5493492639092104, 0.005396207746961571)
```

Note : ClaimedValue vs. P-Value

- As we keep on increasing the claimedValue, P-Value / Probability-Value keeps on decreasing because actual Population mean is getting away from claimedValues

two-sample t test / two-sample Z test

- Two Sample t Test is used to compare the difference of the averages of two independent samples with the claimed value and establish whether the difference is statistically significant or not
- two separate samples, a & b, calculate mean for a & b, and compare the difference of mean with claimed value

- Q1.
- claiming that student of C.Blr. will get 15k more salary than student of C.Kh.
- find the difference, sample/actual difference value = 12500
- (15000 - 12500) check whether difference between claimed and actual value is significant or not

```
x = [3, 7, 6, 5, 9, 8, 2, 3, 5, 4, 6, 7]
y = [9, 6, 8, 10, 6, 4, 7, 8, 9, 10]
# creating two samples
```

Left Tail Test

- H_o : mean(y) - mean(x) >= 2.7
- H_A / $H1$: mean(y) - mean(x) < 2.7 [Left Tail test]

import statsmodels.stats.weightstats

```
import numpy as np
import pandas as pd
# import numpy & pandas

import statsmodels
from statsmodels import stats
from statsmodels.stats import weightstats
# for ztest
```

np.mean(ndarray1) - np.mean(ndarray2)

```
np.mean(y) - np.mean(x)
# actual difference of sample Mean x̄ for two samples to be used to compare
# with claimed mean difference
# and then cestablish whether difference is staistically significant or not
```

```
2.283333333333333
```

statsmodels.stats.weightstats.ztest(Sample1, Sample2, value=ClaimValue, alternative='Sign of Hᴀ')

- `alternative='smaller'` - for left tail Test
- `alternative='larger'` - for right tail Test
- `alternative='two-sided'` - for two tail Test

```
statsmodels.stats.weightstats.ztest(y, x, value=2.7, alternative='smaller')
# alternative='smaller'     - for left tail Test
# alternative='larger'      - for right tail Test
# alternative='two-sided'   - for two tail Test
```

```
(-0.47198804896079605, 0.3184676595008745)
```

```
statsmodels.stats.weightstats.ztest(y, x, value=3.7, alternative='smaller')
```

```
(-1.6047593664667057, 0.054273385712541596)
```

```
statsmodels.stats.weightstats.ztest(y, x, value=4.2, alternative='smaller')
```

```
(-2.1711450252196607, 0.014960105716979144)
```

Note:

- as claimed difference keeps on increasing, P-Value / Probability Value keeps on decreasing

Right Tail Test

- `Hₒ` : mean(y) - mean(x) <= 1.5
- `HA` / `H1` : mean(y) - mean(x) > 1.5 [right tail test]

```
statsmodels.stats.weightstats.ztest(y, x, value=1.5, alternative='larger')
```

```
(0.8873375320462958, 0.1874486022476365)
```

Two Tail Test

- if it was two tail test, `Hₒ` will be equality with some value, and `HA` will be not equal to that value
- `Hₒ` : mean(y) - mean(x) = 1.5
- `HA` / `H1` : mean(y) - mean(x) != 1.5 [two tail test]

```
statsmodels.stats.weightstats.ztest(y, x, value=1.5, alternative='two-sided')
```

Paired t test

- two samples are dependent and paired to each other
- sample size has to be same
- whenever performance/some numbers/parameters need to be compared before and after an event
- in two-sample t test: difference of averages/means of two samples is compared with claimed value
- in paired t test: mean of differences of two samples is compared with claimed value

Q. before & after covid started,

- before people worked from home and after people worked from office

```
before = c(7, 9, 8, 6, 7, 8)
after = c(6, 8, 9, 5, 6, 8)
since people started working from home, their time of delivery has increased by an avg of 0.2min
```

```
- Hₒ : avg(after-before) >= 0.2
- Hᴀ : avg(after-before) < 0.2 ---> left tail test
```

```
# t.test(after, before, mu=0.2, alternative='less', paired=T)
# run in R lang

# t.test(s1, s2, mu=claimedValue, alternative='less', paired=T)
# difference of means is statistically significant as P-Value 0.04786 < 0.05,
# so Hₒ is rejected
```

one-way ANOVA

- ANOVA : ANalysis Of VAriance
- used when we have multiple samples
- ANova uses F-Distribution(Fischer Distribution)
- `Hₒ` : all the pair of means are equal
- `Hᴀ` / `H1` : there is atleast one pair of means where the difference is not equal to zero
- response should follow normal distribution
- variances are equal
- Predictor is attribute/categorical & Response is continuous

```
from google.colab import files
uploaded = files.upload()
# MyData.xlsx

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving MyData.xlsx to MyData.xlsx

```
df = pd.read_excel('MyData.xlsx')
df.head()
```

	Sample	Value
0	A	5
1	A	4
2	A	6
3	A	7
4	A	5
5	B	6
6	B	5
7	B	4
8	B	7
9	B	6
10	C	10
13	C	12

```
import  statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
```

```
# 'sample' is called predictor
# 'value' is called as response
# '~' means depends on
mod1 = ols('Value ~ Sample', data=df).fit()
# ols('colWithResponse ~ ColumnWithPredictor', data=dataFrame).fit()
```

```
tbl = sm.stats.anova_lm(mod1)
tbl
# df : degree of freedom
# degree of freedom for residual = sample size 14 - sample 2 - 1 = 11
# sum_sq : sum of squares
# mean_sq : mean of squares
# F : F-Distribution
# PR(>F) : P-Value from F-distribution
```

	df	sum_sq	mean_sq	F	PR(>F)
Sample	2.0	130.278571	65.139286	37.416822	0.000012
Residual	11.0	19.150000	1.740909	NaN	NaN

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

```
pairwise_tukeyhsd(df.Value, df.Sample)
```

```
<statsmodels.sandbox.stats.multicomp.TukeyHSDResults at 0x7f1ccde72200>
```

```
print(pairwise_tukeyhsd(df.Value, df.Sample))
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     A      B      0.2 0.9689 -2.0538 2.4538  False
     A      C     6.85    0.0  4.4595 9.2405   True
     B      C     6.65    0.0  4.2595 9.0405   True
---------------------------------------------------
```