cdf and ppf

- cdf: Cumulative Distribution Function
- ppf: Percent Point Function
- cdf & ppf are inverse of each other

scipy.stats.norm

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

norm.cdf(Z)

- cdf: Cumulative Distribution Function
- takes Z-Statistics, value on x-axis
- returns area under the curve towards the left side of the mean
- returned area under the curve represents probability / P-Value
- total area under all the curve or total probability = 1

```
norm.cdf(-1)
# cdf takes values from the x-axis, which can be +ve/-ve
# cdf returns area under the curve to the left side of the value on x-axis
```

```
0.15865525393145707
```

norm.ppf(area/prob)

- ppf: Percent Point Function
- takes area under the curve or the probability
- area under the curve as argument can't be -ve
- returns Z-Statistics or Z-Score on X-axis on left side of the mean

```
norm.ppf(0.15865525393145707)
# ppf: Percent Point Function
# Inverse of CDF: Cumulative distribution function
```

```
-1.0
```

```
norm.ppf(0.025)
# takes 0.025 or 2.5%  of area unde rthe curve on the left side of the mean
# returns Z-Statistics on X-axis
```

```
-1.9599639845400545
```

```
norm.ppf(0.975)
# takes 0.025 or 2.5%  of area unde rthe curve on the left side of the mean
# returns Z-Statistics on X-axis
```

```
1.959963984540054
```

```
norm.ppf(0.5)
# takes 0.5 or 50% area under the curve on the left side of the curve
# total area under the curve or total probability = 1
# so Z-Statistics = 0
```

```
0.0
```

Hypothesis Testing

- Null Hypothesis `Hₒ` : No effect exists in the population.
- Alternative Hypothesis `Hᴀ` / `H1` / `H2` : The effect exists in the population.

Steps involved in Hypothesis analysis

Step 1: Establish Null Hypothesis `Hₒ` & Alternative Hypothesis (`Hᴀ` / `H1` / `H2`)

- If Null Hypothesis, `Hₒ` is found to be true, then
    - NO action , NO decision , NO changes

    1. `Right Tail Test` : when error is on right side of Null Hypothesis on number line
        - e.g. delivery time <= 40min, speed limit <= 80kmph
    2. `Left Tail Test` : when error is on left side of Null Hypothesis on number line

        - e.g. voltage <=220

    3. `Two Tail Test`: when error is both sides of an interval

        - e.g. Blood Pressure level <=80 and Blood Pressure level >= 120
        - e.g. Voltage<=180 & voltage>=240

- Null Hypothesis, `Hₒ` will always have an equal to sign

- e.g. salaray promised >=30,000 is Null Hypothesis

Step 2: Establish Confidence Level(C.L.) and the Level of Significance (LoS) for the analysis

- Confidence level (C.L.)

    - default value is >= 0.95 or 95%

- Level of Significane (LoS):

    - `LoS = 1 - C.L.`
    - default value of LoS is <= 0.05 or 5%

Step 3: Choose the appropriate staistical test for performing the analysis

Step 4: use the appropriate functions in python to calculate the P-Value

Step 5: Compare the P-value (which you got from step 4) against the Level of Significance LoS (which you got from Step 2)

- *NOTE:*

    - If `P-value` < `LoS` , then reject `Hₒ`
    - If `P-value` > `LoS` , then don't reject `Hₒ`

- Data used in hypothesis is prone to error as it is not population data, it is sample data

- Statistical Sample = Population parameter + Actual Difference + Chance Variation [not a mathematical equation]

- `Number of possible Hypothesis = Number of variables + 1`

    - for `y = mx + c`, number of variables = 1
    - so, number of possible hypothesis = 2

Types of Error

1. Type 1 Error
2. Type 2 Error

Type 1 Error

- rejecting a Null Hypothesis `Hₒ` which is actually true is Type 1 Error
- rejecting due to limited information
- e.g.: rejecting a good girl / important phone call / some good movie due to limited information

Type 2 Error

- Not rejecting the Null Hypothesis `Hₒ` , which is actually False
- e.g.: not rejecting a bad movie

- `LoS` : max allowed probability of committing Type 1 error
- `P-Value` : actual probability of committing Type 1 Error

- assume maxprob(getting affecting) / LoS = 7%
- if p-value / actual probability is 23%, then we reject `Hₒ`
- if p-value / actual probability is 3%, then we accept `Hₒ`
- if the Sample Statistics and Plain Value(P-Value)/Population Parameter do not meet the criteria, then P-value gives the probability that the difference is just a matter of chance, there is no actual difference and the action can be avoided.

General Note (Hypothesis Testing)

- H 0 : X does not have an impact on Y
  - X : predictor / variables
  - Y : response / result
- if `P-Value < 0.05` (`P<|Z|`)
  - `Hₒ` is rejected
  - X has impact on Y . `y = mx + c`
  - influence of X on Y cannot be ignored
  - It is statistically significant
  - it is not a matter of chance variation

Tools for continuous data

1. Sample t Tests / Sample Z tests
    - one-sample t test / one-sample Z test
    - two-sample t test / two-sample Z test
2. ANOVA tests
    - one-way ANOVA
    - two-way ANOVA
3. one-Sample Variance Test

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 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

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

```

```

```
import numpy as np
import pandas as pd

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


from scipy import stats
from scipy.stats import norm
# for P-value / probability
```

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

```
6.0
```

assuming that, `Hₒ` : Mean(x) is atleast 7.2

Step1: establishing `Hₒ` & `Hᴀ`

=> `Hₒ` : `a >= 7.2` and `Hᴀ` : `a < 7.2`

```
- SS  =   PP  +   ActualDifference    +   ChanceVariation
- 6       7.2                             could be a matter of chance variation
```

Step2: CL = 95% , LoS = 5% = 0.05

Step3: Choose one-sample t test / one-sample Z test to compare a sample mean with the claim value

Step4: use appropriate python function to calculate P-value

statsmodels.stats.weightstats.ztest(list/ndarray, value=Claimed Value, alternative=Sign Of Hᴀ)

```
statsmodels.stats.weightstats.ztest(x, value=7.2, alternative='smaller')
# statsmodels.stats.weightstats.ztest(list/ndarray, value=Claimed Value, alternative=Sign Of Hᴀ)
# returns (TestStatistics, P-Value)
```

```
(-1.9497692171126306, 0.025601816055026008)
```

norm.cdf(testStatistics)

- to cross verify the P-Value with the P-Value we got from ztest() using the TestStatistics value

```
norm.cdf(-1.9497692171126306)
# norm.cdf(testStatistics) returns P-value / Probability
```

```
0.025601816055026008
```

Step5: comapare the P-Value with LoS

- `P-Value(0.025601816055026008)` < `LoS(0.05)`, so we reject the `Hₒ`
- the differenece between 6 and 7.2 is statistically significant, cannot be ignored, so action is required
- Next stepis to take an action and repeat(here repeating with different claimed value)

In next repetition, assuming that, `Hₒ` : Mean(x) is atleast 6.5

Step1: establishing `Hₒ` & `HA`

=> `Hₒ` : a >= 6.5 and `HA` : a < 6.5

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

```
(-0.812403840463596, 0.2082799716383955)
```

- Now, `P-Value(0.2082799716383955)` > `LoS(0.05)`, so we do not reject the `Hₒ`
- But, the differenece between 6 and 6.5 is statistically significant which cannot be ignored, so action is required, and we repeat Step1 to Step5 with different claimed value

scipy.stats.t

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

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

- calculating interval, and then calculating the probability / area under the curve for the interval as Z-statistics on x-axis using CDF method

```
x = stats.t.interval(0.95, 2000-1, 27000, 1000/2000**0.5)
# interval(C.L., dof, x̄, StdErrMean)
# Standard Error of the Mean, StdErrMean = S.D. / sqrt(SampleSize)
# returns (lower_value, higher_value)
x
```

```
(26956.147321103283, 27043.852678896717)
```

scipy.stats.norm

```
import scipy
from scipy import stats
from scipy.stats import norm
norm.cdf(1) - norm.cdf(-1) # 1-σ
```

```
0.6826894921370859
```

norm.cdf(Z)

```
norm.cdf(1)
# area under curve/probability for z-statistics = 1
```

```
0.8413447460685429
```

```
norm.cdf(-1)
# area under curve/probability for z-statistics = -1
```

```
0.15865525393145707
```

```
norm.cdf(26956.147321103283)
# z-statistics is lower end of interval
# area under curve/probability for z-statistics = 26956.147321103283
```

```
1.0
```

```
norm.cdf(27043.852678896717)
# z-statistics is upper end of interval
# area under curve/probability for z-statistics = 27043.852678896717
```

```
1.0
```

```
norm.cdf(26956.147321103283) - norm.cdf(27043.852678896717)
# area under the curve / probability between the intervals
```

```
0.0
```

Note

- Non-parametiric test can be used without considering the distribution