



XIM UNIVERSITY

PYTHON PROJECT

STATISTIC PACKAGE

Submitted by:

Akash Giri

Ashleesha Tripathy

Aaratrika Santra

Subhashree Sahoo

Naren Kumar S

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our professor Dr. Chandan Misra for his able guidance and support in completing this project.

Akash Giri

Ashleesha Tripathy

Aaratrika Santra

Subhashree Sahoo

Naren Kumar S

TABLE OF CONENTS

- Project Introduction
 - About Project
 - Modules used in Project
- System Implementation
 - Hardware Used
 - Software Used
- Functions Built
 - sum() and fsum()
 - mean() , geo_mean() and har_mean()
 - median() and median_low() and median_high()
 - mode()
 - variance() , dev() , svaraiance() and sdev()
 - covariance(), scovariance() and correlation
 - max() and min()
- Bibliography

INTRODUCTION

A statistical package is basically a collection of software routines with a common interface that has been designed to simplify the job of performing statistical analysis and related tasks such as data management. The main thing to remember regarding statistical packages is that, like any computer software, they are only a means to an end.

This project is a genuine attempt to create a statistical package with our knowledge of python and its modules.

Modules used in the project :

➤ Math Module:

- This module provides access to the mathematical functions defined by the C standard.

➤ Numpy Module:

- NumPy is a Python library used for working with arrays. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.
- Used in the project to carry out vectorized operation.

System implementation

1. HARDWARE USED:-

Model: DELL inspiron 15300

Manufacture: DELL Technologies

Model: DELL INSPIRON LAPTOP

Category: Laptop Computer

[View basic information about your computer](#)

Windows edition

Windows 10 Home Single Language

© 2020 Microsoft Corporation. All rights reserved.

System

Processor: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

Installed memory (RAM): 8.00 GB (7.77 GB usable)

System type: 64-bit Operating System, x64-based processor

Pen and Touch: No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

Computer name: DESKTOP-UMH81H1

Full computer name: DESKTOP-UMH81H1

Computer description:

Workgroup: WORKGROUP

Windows activation

Windows is activated [Read the Microsoft Software License Terms](#)

Product ID: 00327-35899-57530-AAOEM

2.SOFTWARE USED: -

❖ ANACONDA (Python distribution):-

Anaconda is a distribution of the python and R programming language for scientific computing (data science, machine learning application , large scale data processing, predictive analysis, etc.),that aims to simplify package management and deployment. The distribution includes data-science packages suitable for windows , Linux , and mac os .It is developed and maintained by Anaconda ,inc., which was founded by Peter wang and Travis Oliphant in 2012. As an Anaconda. Inc., product ,it is known as Anaconda Distribution or Anaconda individual edition , while other products from the company are anaconda team edition and anaconda enterprise edition, both of which are not free.

Anaconda



Developer(s)	Anaconda, Inc. ^[1] (previously Continuum Analytics) ^[2]
Initial release	0.8.0 ^[3] /17 July 2012; 8 years ago
Stable release	2020.11 / 19 November 2020; 52 days ago ^[4]
Written in	Python
Operating system	Windows, macOS, Linux
Type	Programming language, machine learning, data science
License	non-commercial use ^[5]
Website	www.anaconda.com 

fsum()

Python `fsum()` function is similar to `sum()` function, returns the sum.

```
1 - def fsum(a):
2     s = 0
3     for i in a:
4         s = s + i
5         s = (float)(s)
6         s = s + 1
7     print(s)
8
9
10 lst=[2.3,3.4,4.9,5.2,6.3,7.7]
11 fsum(lst)
12 fsum([.1,.1,.1,.1,.1,.1,.1,.1,.1,.1,.1])
```

35.800000000000004
10.999999999999998

However, `fsum()` returns the value keeping higher floating accuracy. It takes an iterable as its argument. It returns an accurate floating point sum of values in the iterable. Avoids loss of precision by tracking multiple intermediate partial sums

Fsum() and sum()

`fsum()` is included in Python `math` module so we have to import it before using. Whereas `sum()` is part of built-in functions of core Python so no need to import any library.

`fsum()` returns always float dtype. However `sum()` returns the same dtype of input number.

mean()

To find the mean of a range of numbers.

Suppose $x_1, x_2, x_3, \dots, x_n$ be n observations of a data set, then the mean of these values is:

$$\bar{x} = \frac{\sum x_i}{n}$$

Here,

x_i = i th observation, $1 \leq i \leq n$

$\sum x_i$ = Sum of observations

n = Number of observations

```
1 def mean(a):
2     s = 0
3     for i in a:
4         s = s + i
5     d = (float)(s/len(a))
6     print(d)
7
8 lst=[2.3,3.4,4.9,5.2,6.3,7.7]
9 mean(lst)
```



4.966666666666667

geo_mean()

Geometric Mean (GM) is the average value or mean which signifies the central tendency of the set of numbers by finding the product of their values.

$$G.M = \sqrt[n]{x_1 \times x_2 \times \dots x_n}$$

or

$$G.M = (x_1 \times x_2 \times \dots x_n)^{\frac{1}{n}}$$

```
1 import math
2 def geo_mean(a):
3     n = len(a)
4     s = 1
5     for i in a:
6         s = s*i
7     k = pow(s,1/n)
8     print(k)
9
10
11 lst=[4,16,9]
12 geo_mean(lst)
```

```
8.320335292207616
```

The term central tendency refers to the middle, or typical, value of a set of data, which is most commonly measured by using the three m's: **mean, median, and mode**.

har_mean()

The **Harmonic Mean (HM)** is defined as the reciprocal of the average of the reciprocals of the data values.. It is based on all the observations, and it is rigidly defined.

If $x_1, x_2, x_3, \dots, x_n$ are the individual items up to n terms, then,

$$\text{Harmonic Mean, HM} = n / [(1/x_1) + (1/x_2) + (1/x_3) + \dots + (1/x_n)]$$

$$\text{Arithmetic Mean} = (a_1 + a_2 + a_3 + \dots + a_n) / n$$

$$\text{Harmonic Mean} = n / [(1/a_1) + (1/a_2) + (1/a_3) + \dots + (1/a_n)]$$

$$\text{Geometric Mean} = \sqrt[n]{a_1 \cdot a_2 \cdot a_3 \dots a_n}$$

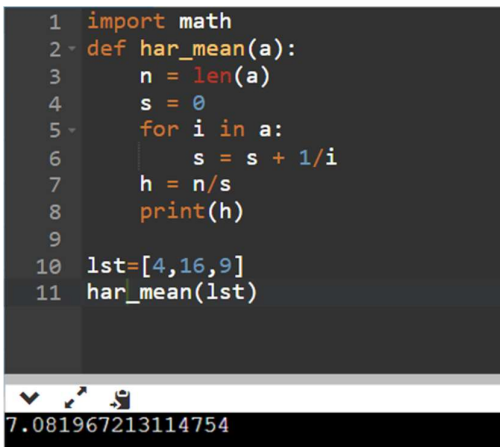
If G is the geometric mean, H is the harmonic mean, and A is the arithmetic mean, then the relationship between them is given by:

$$G = \sqrt{AH}$$

Or

$$G^2 = A.H$$

```
1 import math
2 def har_mean(a):
3     n = len(a)
4     s = 0
5     for i in a:
6         s = s + 1/i
7     h = n/s
8     print(h)
9
10 lst=[4,16,9]
11 har_mean(lst)
```



7.081967213114754

median()

Median, in statistics, is the middle value of the given list of data when arranged in an order. The arrangement of data or observations can be made either in ascending order or descending order.

Odd Number of Observations

If the total number of observations given is odd, then the formula to calculate the median is:

$$\text{Median} = \left(\frac{n+1}{2}\right)^{\text{th}} \text{ term}$$

where n is the number of observations

Even Number of Observations

If the total number of observation is even, then the median formula is:

$$\text{Median} = \frac{\left(\frac{n}{2}\right)^{\text{th}} \text{ term} + \left(\frac{n}{2} + 1\right)^{\text{th}} \text{ term}}{2}$$

where n is the number of observations

```
1 import math
2 def median(a):
3     b = sorted(a)
4     print(b)
5     n = len(a)
6     if n%2 != 0:
7         p = (int)((n+1)/2) - 1
8         print(b[p])
9     else:
10        k = (int)(n/2)-1
11        l = k + 1
12        m = b[k] + b[l]
13        p = m/2
14        print(p)
15
16 lst=[4,16,9,5,6,7,8,8,8,4,4,4]
17 median(lst)
```

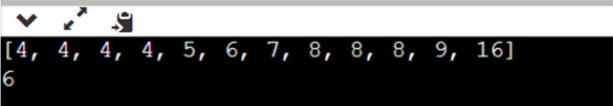
```
▼ ↗ 📄
[4, 4, 4, 4, 5, 6, 7, 8, 8, 8, 9, 16]
6.5
```

median_low()

`median_low()` method calculates the low median of the given data set. This method also sorts the data in ascending order before calculating the low median.

Note: If the number of data values is odd, it returns the exact middle value. If the number of data values is even, it returns the smaller of the two middle values.

```
1 import math
2 def median_low(a):
3     b = sorted(a)
4     print(b)
5     n = len(a)
6     if n%2 != 0:
7         p = (int)((n+1)/2) - 1
8         print(b[p])
9     else:
10        k = (int)(n/2)-1
11        l = k + 1
12        m = b[k]
13        n = b[l]
14        if m > n :
15            p = n
16        else:
17            p = m
18        print(p)
19
20 lst=[4,16,9,5,6,7,8,8,8,4,4,4]
21 median_low(lst)
```



[4, 4, 4, 4, 5, 6, 7, 8, 8, 8, 9, 16]
6

median_high()

`median_high()` method calculates the high median of the given data set. This method also sorts the data in ascending order before calculating the high median.

Note: If the number of data values is odd, it returns the exact middle value.

```
1 import math
2 def median_high(a):
3     b = sorted(a)
4     print(b)
5     n = len(a)
6     if n%2 != 0:
7         p = (int)((n+1)/2) - 1
8         print(b[p])
9     else:
10        k = (int)(n/2)-1
11        l = k + 1
12        m = b[k]
13        n = b[l]
14        if m > n :
15            p = m
16        else:
17            p = n
18        print(p)
19
20
21 lst=[4,16,9,5,6,7,8,8,8,4,4,4]
22 median_high(lst)
```

```
[4, 4, 4, 4, 5, 6, 7, 8, 8, 8, 9, 16]
7
```

mode()

Value or a number that appears most frequently in a data set is a mode.

```
1 def mode(a):
2     b = set(a)
3     #print(b)
4     d = {}
5     for i in b:
6         #c = i
7         count = 0
8         for j in a:
9             if i == j:
10                count = count + 1
11            d[i] = count
12        #print(d)
13
14        a = max(d.values())
15        value = {i for i in d if d[i]== a}
16        print(value)
17
18 lst=[2,2,3,3,2,2,2,6,5,7,8,8,4,4]
19 mode(lst)
```

```
{2}
```

variance()

The term variance refers to a statistical measurement of the spread between numbers in a data set.

Variance is calculated by using the following formula:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{N}$$

where:

x_i = Each value in the data set

\bar{x} = Mean of all values in the data set

N = Number of values in the data set

```
1
2 def variance(a): #population variance
3     s = 0
4     for i in a:
5         s += i
6     mean = s/len(a)
7     d = 0
8     for i in a:
9         ab = pow(abs(i-mean),2)
10        d += ab
11    std = d/len(a)
12    print(std)
13
14 lst=[4,16,9,6,6,7,7,8,8,8,8]
15 variance(lst)
```

7.576388888888887

Svariance()

```
1 def svariance(a): #sample variance
2     s = 0
3     for i in a:
4         s += i
5     mean = s/len(a)
6     d = 0
7     for i in a:
8         ab = pow(abs(i-mean),2)
9         d += ab
10    f = len(a)
11    std = d/(f - 1)
12    print(std)
13
14 lst=[4,16,9,6,6,7,7,8,8,8,8]
15 svariance(lst)
```

8.265151515151514

dev()

Standard Deviation is a measure which shows how much variation (such as spread, dispersion, spread,) from the mean exists. The standard deviation indicates a “typical” deviation from the mean.

Standard Deviation Formula

The population standard deviation formula is given as:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (X_i - \mu)^2}$$

Here,

σ = Population standard deviation

N = Number of observations in population

X_i = ith observation in the population

μ = Population mean

```
1 import math
2 def dev(a): #population standard deviation
3     s = 0
4     for i in a:
5         s += i
6     mean = s/len(a)
7     d = 0
8     for i in a:
9         ab = pow(abs(i-mean),2)
10        d += ab
11    std = d/len(a)
12    dev = math.sqrt(std)
13    print(dev)
14
15 lst=[4,16,9,6,6,7,7,8,8,8,8]
16 dev(lst)
```

2.75252409415229

sdev()

```
1 import math
2 def sdev(a): # sample std deviation
3     s = 0
4     for i in a:
5         s += i
6     mean = s/len(a)
7     d = 0
8     for i in a:
9         ab = pow(abs(i-mean),2)
10        d += ab
11    f = len(a)
12    std = d/(f-1)
13    dev = math.sqrt(std)
14    print(dev)
15
16 lst=[4,16,9,6,6,7,7,8,8,8,8]
17 sdev(lst)
```

2.8749176536296677

covariance()

Covariance is a measure of the relationship between two random variables and to what extent, they change together.

Population Covariance Formula

$$\text{Cov}(X,Y)=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{N}$$

x_i = data value of x

y_i = data value of y

\bar{x} = mean of x




\bar{y} = mean of y

N = number of data values.

```

1 import numpy as np
2 def covariance(x,y): #population covariance
3     s = 0
4     for i in x:
5         s += i
6     mx = s/len(x)
7     print(mx)
8
9     s=0
10    for i in y:
11        s += i
12    my = s/len(y)
13    print(my)
14
15    p = 0
16    a = np.array(x) - mx
17    #print(a)
18    b = np.array(y) - my
19    #print(b)
20    c = a*b
21    #print(c)
22    for i in c:
23        p = p+i
24
25    #d = len(x)
26    #print(d)
27    prod = p/ len(x)
28    print(prod)
29

```

7.916666666666667
5.416666666666667
-0.3819444444444444

scovariance()

Sample Covariance Formula

$$\text{Cov}(X,Y)=\frac{\sum (x_i-\bar{x})(y_i-\bar{y})}{N-1}$$

```
1 import numpy as np
2 def scovariance(x,y): #sample covariance
3     s = 0
4     for i in x:
5         s += i
6     mx = s/len(x)
7     print(mx)
8
9     s=0
10    for i in y:
11        s += i
12    my = s/len(y)
13    print(my)
14
15    p = 0
16    a = np.array(x) - mx
17    b = np.array(y) - my
18    c = a*b
19
20    for i in c:
21        p = p+i
22
23    d = len(x)
24    prod = p/(d-1)
25    print(prod)
26    s = 0
27    for i in x:
28        s += i
29    mx = s/len(x)
30    print(mx)
31
32    s=0
33    for i in y:
34        s += i
35    my = s/len(y)
36    print(my)
37
38    p = 0
39    a = np.array(x) - mx
40    #print(a)
41    b = np.array(y) - my
42    #print(b)
43    c = a*b
44    #print(c)
45    for i in c:
46        p = p+i
47
48    d = len(x)
49    #print(d)
50    prod = p/ d-1
51    print(prod)
52
53    lst=[4,16,9,6,6,7,7,8,8,8,8]
54    lst2=[2,3,4,6,7,1,3,5,7,9,9]
55    scovariance(lst,lst2)
```

```
7.916666666666667
5.416666666666667
-0.41666666666666663
7.916666666666667
5.416666666666667
-1.3819444444444444
```

correlation()

$$\text{Correlation} = \frac{\text{Cov}(x,y)}{\sigma_x \sigma_y}$$

```
1 import numpy as np
2 import math
3 def correlation(x,y): #correlation
4     s = 0
5     for i in x:
6         s += i
7     mx = s/len(x)
8     #print(mx)
9
10    s=0
11    for i in y:
12        s += i
13    my = s/len(y)
14    #print(my)
15
16    a = np.array(x) - mx
17    b = np.array(y) - my
18    c = a*b
19    p = 0
20    for i in c:
21        p = p+i
22    #print(p)
23    k = pow(a,2)
24    g = 0
25    for i in k:
26        g = g+i
27    #print(g)
28
29    l = pow(b,2)
30
31    h = 0
32    for i in l:
33        h = h+i
34    #print(h)
35
36    f = g*h
37    #print(f)
38
39    m = math.sqrt(f)
40
41    cr = p /m
42    print(cr)
43
44 lst=[4,16,9,6,6,7,7,8,8,8,8]
45 lst2=[2,3,4,6,7,1,3,5,7,9,9]
46 correlation(lst,lst2)
```

-0.05097627587875791

maxvalue()

The maxvalue() function returns the item with the highest value, or the item with the highest value in an iterable.

minvalue()

The minvalue() function returns the item with the lowest value, or the item with the lowest value in an iterable.

```
def maxvalue(a):
    maxvalue = a[0]
    for i in range(0, len(a), 1):
        if maxvalue < a[i]:
            maxvalue = a[i]
    print(maxvalue)

def minvalue(a):
    minvalue = a[0]
    for i in range(0, len(a), 1):
        if minvalue > a[i]:
            minvalue = a[i]
    print(minvalue)

a = [1, 2, 3, 4, 5, 6]
maxvalue(a)
minvalue(a)
```

```
In [1]: runfile('C:/Users/Dell/Desktop/Python project/statistic_package.py', wdir='C:/Users/
Dell/Desktop/Python project')
6
1
```

PROJECT SUMMARY

We can use the import keyword to make code in one module available in another module.

In this case , we can import the module as :

“ Import statistical_package as pk “

Here pk is alias name for statistical_package module.

BIBLIOGRAPHY:-

- Python Data Science Handbook
- Python documentation:
<https://docs.python.org/3/library/statistics.html>
- Numpy Documentation