Lisbon School of Economics & Management
Universidade de Lisboa

U LISBOA
UNIVERSIDADE DE LISBOA

Carlos J. Costa

STATISTICS

Learning Goals

- Students should use the main libraries to present descriptive statistics.
- Students should know how to perform normality test using Python libraries.
- Students should know how to perform parametric statistical hypothesis Tests using Python libraries.

Statistics module

- This module provides functions for calculating mathematical statistics of numeric (Real-valued) data.
- The module is not intended to be a competitor to third-party libraries (e.g. NumPy, SciPy, Minitab, SAS and Matlab).
- https://docs.python.org/3.7/library/statistics.html

Averages and measures of central location¶

mean()	Arithmetic mean (“average”) of data.
harmonic_mean()	Harmonic mean of data.
median()	Median (middle value) of data.
median_low()	Low median of data.
median_high()	High median of data.
median_grouped()	Median, or 50th percentile, of grouped data.
mode()	Mode (most common value) of discrete data.

```
import statistics

a=[1,2,3,9,2,5,1,3,4,1,4,2,5,3,4,5,6]

print(statistics.mode(a))
print(statistics.mean(a))
print(statistics.median(a))
```

```
1
3.5294117647058822
3
```

Measures of spread

pstdev()	Population standard deviation of data.
pvariance()	Population variance of data.
stdev()	Sample standard deviation of data.
variance()	Sample variance of data.

Numpy

Function	Numpy
Min	np.min()
Max	np.max()
Mean	np.mean()
Median	np.median()
Standard deviation	np.std()

```
import numpy as np
arrayA=np.array([1,2,3,4,5,6,7,8,9,1,2,3,3])
print(arrayA.min())
print(arrayA.max())
print(arrayA.mean())
```

```
1
9
4.153846153846154
```

```
import numpy as np
b=[1,2,3,4,5,6,7,8,9,1,2,3,3]
print(np.mean(b))
```

```
4.153846153846154
```

- Data analysis and manipulation Tool
- Fast, powerful, flexible and easy to use
- Open-source (BSD-3-Clause License)
- It is built on top of the Python programming language.

```
import pandas as pd
students={'name':['ALEKSANDRA','KAROLINA','MARC-PASCAL','VICTOIRE'],
          'country':['FR','FR','PL','PL'],
          'age':[20,20,22,21],
         'grade':[17,18,16,19] }
df=pd.DataFrame(students)
df.describe()
```

]:

	age	grade
count	4.000000	4.000000
mean	20.750000	17.500000
std	0.957427	1.290994
min	20.000000	16.000000
25%	20.000000	16.750000
50%	20.500000	17.500000
75%	21.250000	18.250000
max	22.000000	19.000000

- This module contains a
- https://scipy.org - SciPy stack
- https://scipy.org/scipylib/index.html - SciPy library
- Large number of probability distributions
- Large number of statistical functions

Statistical tests

ttest_1samp(a, popmean[, axis])	Calculates the T-test for the mean of ONE group of scores.
ttest_onesamp(a, popmean[, axis])	Calculates the T-test for the mean of ONE group of scores.
ttest_ind(a, b[, axis, equal_var])	Calculates the T-test for the means of TWO INDEPENDENT samples of scores.
ttest_rel(a, b[, axis])	Calculates the T-test on TWO RELATED samples of scores, a and b.
chisquare(f_obs[, f_exp, ddof, axis])	Calculate a one-way chi-square test.
kstest(data1, data2[, args, alternative, mode])	**Parameters:**
ks_2samp(data1, data2[, alternative, mode])	Computes the Kolmogorov-Smirnov test on two samples.
ks_1samp(x, cdf[, args, alternative, mode])	Computes the Kolmogorov-Smirnov test on one sample of masked values.
ks_twosamp(data1, data2[, alternative, mode])	Computes the Kolmogorov-Smirnov test on two samples.
mannwhitneyu(x, y[, use_continuity])	Computes the Mann-Whitney statistic
rankdata(data[, axis, use_missing])	Returns the rank (also known as order statistics) of each data point along the given axis.
kruskal(*args)	Compute the Kruskal-Wallis H-test for independent samples
kruskalwallis(*args)	Compute the Kruskal-Wallis H-test for independent samples
friedmanchisquare(*args)	Friedman Chi-Square is a non-parametric, one-way within-subjects ANOVA.
brunnermunzel(x, y[, alternative, distribution])	Computes the Brunner-Munzel test on samples x and y
skewtest(a[, axis])	Tests whether the skew is different from the normal distribution.
kurtosistest(a[, axis])	Tests whether a dataset has normal kurtosis
normaltest(a[, axis])	Tests whether a sample differs from a normal distribution.

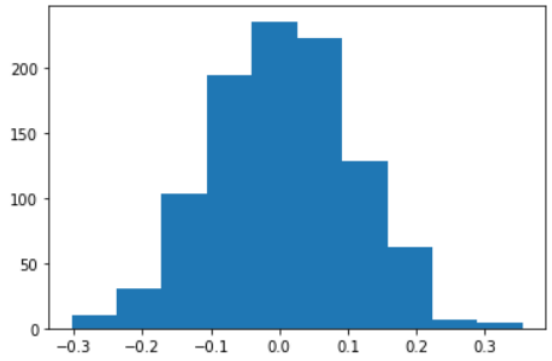
https://docs.scipy.org/doc/scipy/reference/stats.html

Generating a Normal Distribution

```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 0, 0.1 # mean and standard deviation
a = np.random.normal(mu, sigma, 1000)

plt.hist(a)
plt.show()
```



Normality test

- H0: p>0.05 (Normal Distribution)
- H1: p<0.05

```
from scipy import stats

stat, p = stats.normaltest(a)

if p < 0.05:
    print("The null hypothesis can be rejected")
else:
    print("The null hypothesis cannot be rejected")
```

```
The null hypothesis cannot be rejected
```

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.normaltest.html#scipy.stats.normaltest

D'Agostino, R. B. (1971), "An omnibus test of normality for moderate and large sample size", Biometrika, 58, 341-348
D'Agostino, R. and Pearson, E. S. (1973), "Tests for departure from normality", Biometrika, 60, 613-622

Normality test

- H0: p>0.05 (Normal Distribution)
- H1: p<0.05

```
from scipy import stats

stat, p = stats.shapiro(a)

if p < 0.05:
    print("The null hypothesis can be rejected")
else:
    print("The null hypothesis cannot be rejected")
```

```
The null hypothesis cannot be rejected
```

Shapiro, S. S. & Wilk, M.B (1965). An analysis of variance test for normality (complete samples), Biometrika, Vol. 52, pp. 591-611.

Normality test

- H0: p>0.05 (Normal Distribution)
- H1: p<0.05

```
from scipy import stats

stat, p = stats.shapiro(a)

if p
    p
else:
    p
```

```
The r
```

```
from scipy.stats import shapiro

stat, p = shapiro(a)

if p < 0.05:
    print("The null hypothesis can be rejected")
else:
    print("The null hypothesis cannot be rejected")
```

```
The null hypothesis cannot be rejected
```

Student's t-test

- H0: p>0.05 the means of the samples are equal.
- H1: p<0.05 .

```
from scipy import stats

sample1 = [2, 4, 6, 7, 9, 9, 9, 9, 1, 2]
sample2 = [2, 4, 6, 7, 9, 9, 9, 9, 1, 2]

stat, p = stats.ttest_ind(sample1, sample2, equal_var = False)

if p < 0.05:
    print('The null hypothesis can be rejected')
else:
    print('The null hypothesis cannot be rejected')
```

```
The null hypothesis cannot be rejected
```

This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values.

Paired Student's t-test

- H0: p>0.05 the means of the samples are equal.
- H1: p<0.05 .

```
from scipy import stats

sample1 = [2, 4, 6, 7, 9, 9, 9, 9, 1, 2]
sample2 = [2, 4, 6, 7, 9, 9, 9, 9, 1, 2]

stat, p = stats.ttest_rel(sample1, sample2)

if p < 0.05:
    print('The null hypothesis can be rejected')
else:
    print('The null hypothesis cannot be rejected')
```

```
The null hypothesis cannot be rejected
```

This is a two-sided test for the null hypothesis that 2 related or repeated samples have identical average (expected) values.

Analysis of Variance Test (ANOVA)

- H0: p>0.05 the means of the samples are equal.
- H1: p<0.05

```
from scipy import stats

sample1 = [24, 4, 6, 7, 9, -79, 9, 9, 1, 72]
sample2 = [22, 0, 76, 0, 9, 0, 0, 9, 1, 0]
sample3 = [26, 64, 67, -77, 9, 9, 9, 9, 71, 2]

stat, p = stats.f_oneway(sample1, sample2, sample3)

if p < 0.05:
    print('The null hypothesis can be rejected')
else:
    print('The null hypothesis cannot be rejected')
```

```
The null hypothesis cannot be rejected
```

The one-way ANOVA tests the null hypothesis that two or more groups have the same population mean.
The test is applied to samples from two or more groups, possibly with differing sizes.