

**School of Built Environment, Engineering and Computing**



**Week 3 : Inferential Statistics 2:**

**Correlation and Chi Square**

**By**

**Prof. Ah-Lian Kor**

1. **Correlation**

Correlation measures the strength of the relationship between two variables (numeric or continuous). We shall use numpy, scipy, and pandas.

**Types of correlation**

Between 2 columns

* 1. **Use pandas.Dataframe.corr() and pandas.Series.corr() to find the correlation between two columns. It calculates the Pearson Correlation Coefficient for the 2 columns or more.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | a | B | c | D | e |
| 0 | 10 | 20 | 10 | 15 | 140 |
| 1 | 20 | 45 | 45 | 25 | 120 |
| 2 | 30 | 55 | 67 | 33 | 55 |
| 3 | 40 | 78 | 99 | 43 | 65 |
| 4 | 50 | 82 | 98 | 54 | 44 |
| 5 | 60 | 88 | 100 | 61 | 30 |
| 6 | 70 | 95 | 120 | 71 | 3 |

We shall assign values to the columns in the program. This means we are not going to read it in from a file.

**Example 1.1.1: Find Pearson correlation coefficient (r) between 2 columns - use pandas.DataFrame**

**Type the following**

#Pandas Dataframe

#Simple Correlation

import pandas as pd

import numpy as np

dataset = {'a': [10, 20, 30, 40, 50, 60, 70],

'b': [20, 45, 55, 78, 82, 88, 95],

'c': [10, 45, 67, 99, 98, 100, 120],

'd': [15, 25, 33, 43, 54, 61, 71],

'e': [140, 120, 55, 65, 44, 30, 3]

}

df = pd.DataFrame(data=dataset)

column\_0 = df["a"]

column\_1 = df["b"]

column\_2 = df["c"]

column\_3 = df["d"]

column\_4 = df["e"]

#Pearson Correlation

correlation1 = column\_0.corr(column\_1)

correlation2 = column\_1.corr(column\_2)

correlation3 = column\_2.corr(column\_3)

correlation4 = column\_3.corr(column\_4)

print("Correlation between a and b is: ", correlation1)

print("\n")

print("Correlation between b and c is: ", correlation2)

print("\n")

print("Correlation between c and d is: ", correlation3)

print("\n")

print("Correlation between d and e is: ", correlation4)

print("\n")

**Example 1.1.2: Find Pearson correlation coefficients (r) between all the columns and display as a correlation matrix - use pandas.Dataframe.**

**Type the following:**

#Pandas Dataframe

#Correlation Matrix

import pandas as pd

import numpy as np

dataset = {'a': [10, 20, 30, 40, 50, 60, 70],

'b': [20, 45, 55, 78, 82, 88, 95],

'c': [10, 45, 67, 99, 98, 100, 120],

'd': [15, 25, 33, 43, 54, 61, 71],

'e': [140, 120, 55, 65, 44, 30, 3]

}

df = pd.DataFrame(data=dataset)

correlation\_df = df.corr()

print("\n")

print("Correlation Matrix for d to e: ")

print("\n")

print(correlation\_df)

print("\n")

#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

# Others are: kendall(Kendall Tau correlation coefficient)

#spearman (Spearman rank correlation)

correlation\_df2 = df.corr(method = 'pearson')

print("\n")

print("Pearson Standard Correlation Coefficient")

print(correlation\_df2)

print("\n")

**Example 1.1.3. Use pandas.Series.corr() to find the correlation coefficient (r) between two columns.**

**Type the following:**

**#Panda Series**

**#Simple Correlation**

**import pandas as pd**

**import numpy as np**

**a = pd.Series([10, 20, 30, 40, 50, 60, 70])**

**b = pd.Series([20, 45, 55, 78, 82, 88, 95])**

**c = pd.Series([10, 45, 67, 99, 98, 100, 120])**

**d = pd.Series([15, 25, 33, 43, 54, 61, 71])**

**e = pd.Series([140, 120, 55, 65, 44, 30, 3])**

**column\_0 = a**

**column\_1 = b**

**column\_2 = c**

**column\_3 = d**

**column\_4 = e**

**#Pearson Correlation**

**correlation1 = column\_0.corr(column\_1).round(decimals=3)**

**correlation2 = column\_1.corr(column\_2).round(decimals=3)**

**correlation3 = column\_2.corr(column\_3).round(decimals=3)**

**correlation4 = column\_3.corr(column\_4).round(decimals=3)**

**print("Correlation between a and b is: ", correlation1)**

**print("\n")**

**print("Correlation between b and c is: ", correlation2)**

**print("\n")**

**print("Correlation between c and d is: ", correlation3)**

**print("\n")**

**print("Correlation between d and e is: ", correlation4)**

**print("\n")**

**Sources**

<https://www.kite.com/python/answers/how-to-find-the-correlation-between-two-pandas-dataframe-columns-in-python>

<https://medium.com/analytics-vidhya/basic-of-correlations-and-using-pandas-and-scipy-for-calculating-correlations-2d16c2bd6af0>

**Syntax for pandas correlation**

[Use pandas.Series.corr()](https://www.kite.com/python/answers/how-to-find-the-correlation-between-two-pandas-dataframe-columns-in-python" \l "use-series-corr)

[Use pandas.DataFrame.corr()](https://www.kite.com/python/answers/how-to-find-the-correlation-between-two-pandas-dataframe-columns-in-python" \l "use-df-corr)

**1.2 Heatmaps for correlation**

We shall use seaborn.heatmap. (Note: seaborn is for data visualisation).

Heatmap for Correlation Matrix (without mask-full rectangle and label).

**Example 1.2.1 - Heatmap without label and mask**

Make sure the pima-indians-diabetes.xlsx file is in the Jupyter Notebook DAV folder. Note the Pearson correlation matrix is only for two continuous variables. We are going to create a correlation matrix for all continuous variables.

**Type the following:**

#Seaborn for correlation heatmaps without mask

#adapted from https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

#read in the pima-indians-diabetese.xlsx dataset

dataset = pd.read\_excel('pima-indians-diabetes.xlsx')

df = pd.DataFrame(dataset, columns= ['pregNo', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age'])

print (df)

#create a correlation matrix for all the column sets except the target variable

correlation = df.corr()

print("\n")

print("Pearson Standard Correlation Coefficient Matrix")

print(correlation)

print("\n")

#Set up matplotlib figure

f, ax = plt.subplots(figsize=(11, 9))

#Set up a seaborn heatmap

#seaborne aesthetics https://seaborn.pydata.org/tutorial/aesthetics.html

sns.set\_style("white")

# Generate a custom diverging colormap

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio

sns.heatmap(correlation, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5})

**Example 1.2.2 - Heatmap without label and with mask**

**Type the following and compare the output with the output in Example 1.2.1.**

#Seaborn for correlation heatmaps with mask

#adapted from https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

#read in the pima-indians-diabetese.xlsx dataset

dataset = pd.read\_excel('pima-indians-diabetes.xlsx')

df = pd.DataFrame(dataset, columns= ['pregNo', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age'])

print (df)

#create a correlation matrix for all the column sets except the target variable

correlation = df.corr()

print("\n")

print("Pearson Standard Correlation Coefficient Matrix")

print(correlation)

print("\n")

# Generate a mask for the upper triangle

mask = np.triu(np.ones\_like(correlation, dtype=bool))

#Set up matplotlib figure

f, ax = plt.subplots(figsize=(11, 9))

#Set up a seaborn heatmap

#seaborne aesthetics https://seaborn.pydata.org/tutorial/aesthetics.html

sns.set\_style("white")

# Generate a custom diverging colormap

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio

sns.heatmap(correlation, mask = mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5})

**Example 1.2.3 Heatmap with label (or annotation) and mask**

**Type the following:**

#Seaborn for correlation heatmaps with labels or annotations

#adapted from https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

#read in the pima-indians-diabetese.xlsx dataset

dataset = pd.read\_excel('pima-indians-diabetes.xlsx')

df = pd.DataFrame(dataset, columns= ['pregNo', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age'])

print (df)

#create a correlation matrix for all the column sets except the target variable

correlation = df.corr()

print("\n")

print("Pearson Standard Correlation Coefficient Matrix")

print(correlation)

print("\n")

# Generate a mask for the upper triangle

mask = np.triu(np.ones\_like(correlation, dtype=bool))

#Set up matplotlib figure

f, ax = plt.subplots(figsize=(11, 9))

#Set up a seaborn heatmap

#seaborne aesthetics https://seaborn.pydata.org/tutorial/aesthetics.html

sns.set\_style("white")

# Generate a custom diverging colormap

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

#Turn on annotations for correlation values

#annot = True

#annot\_kws = {"size": 12}

# Draw the heatmap with the mask and correct aspect ratio

heatmap = sns.heatmap(correlation, mask = mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5},

annot = True, annot\_kws = {"size": 12})

#Add ticks for labels

sns.set\_style({'xtick.bottom': True}, {'ytick.left': True})

#Export heatmap as an image

heatmap.get\_figure().savefig('heatmap\_diabetes.png', bbox\_inches='tight')

**Sources:**

<https://heartbeat.fritz.ai/seaborn-heatmaps-13-ways-to-customize-correlation-matrix-visualizations-f1c49c816f07>

Detailed Explanation

<https://www.kdnuggets.com/2019/07/annotated-heatmaps-correlation-matrix.html>

Plot diagonal pairwise correlation matrix using seaborn.

<https://seaborn.pydata.org/examples/many_pairwise_correlations.html>

With annotations or labels

<https://www.kdnuggets.com/2019/07/annotated-heatmaps-correlation-matrix.html>

**1.3 How to calculate the importance of features?**

Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. This is important when you want to improve the accuracy of a model. We shall use mobile.csv. The data dictionary is as follows:

Citation: <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=You%20can%20get%20the%20feature,feature%20towards%20your%20output%20variable>.

**Description of variables in the above file**

battery\_power: Total energy a battery can store in one time measured in mAh

blue: Has Bluetooth or not

clock\_speed: the speed at which microprocessor executes instructions

dual\_sim: Has dual sim support or not

fc: Front Camera megapixels

four\_g: Has 4G or not

int\_memory: Internal Memory in Gigabytes

m\_dep: Mobile Depth in cm

mobile\_wt: Weight of mobile phone

n\_cores: Number of cores of the processor

pc: Primary Camera megapixels

px\_height

Pixel Resolution Height

px\_width: Pixel Resolution Width

ram: Random Access Memory in MegaBytes

sc\_h: Screen Height of mobile in cm

sc\_w: Screen Width of mobile in cm

talk\_time: the longest time that a single battery charge will last when you are

three\_g: Has 3G or not

touch\_screen: Has touch screen or not

wifi: Has wifi or not

price\_range: This is the target variable with a value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

<https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=You%20can%20get%20the%20feature,feature%20towards%20your%20output%20variable>.

**Example 1.3.1 Create a heatmap with label and mask for mobile.csv**

**Type the following:**

#Seaborn for correlation heatmaps with labels or annotations

#adapted from https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

#read in the pima-indians-diabetese.xlsx dataset

dataset = pd.read\_csv('mobile.csv')

df = pd.DataFrame(dataset)

print (df)

#create a correlation matrix for all the column sets except the target variable

correlation = df.corr()

print("\n")

print("Pearson Standard Correlation Coefficient Matrix")

print(correlation)

print("\n")

# Generate a mask for the upper triangle

mask = np.triu(np.ones\_like(correlation, dtype=bool))

#Set up matplotlib figure

f, ax = plt.subplots(figsize=(25, 15))

#Set up a seaborn heatmap

#seaborne aesthetics https://seaborn.pydata.org/tutorial/aesthetics.html

sns.set\_style("white")

# Generate a custom diverging colormap

cmap = sns.diverging\_palette(230, 20, as\_cmap=True)

#Turn on annotations for correlation values

#annot = True

#annot\_kws = {"size": 12}

# Draw the heatmap with the mask and correct aspect ratio

heatmap = sns.heatmap(correlation, mask = mask, cmap=cmap, vmax=.3, center=0,

square=True, linewidths=.5, cbar\_kws={"shrink": .5},

annot = True, annot\_kws = {"size": 12})

#Add ticks for labels

sns.set\_style({'xtick.bottom': True}, {'ytick.left': True})

#Export heatmap as an image

heatmap.get\_figure().savefig('mobile.png', bbox\_inches='tight')

**Example 1.3.2 Choose the top 10 variables for the target variable (I.e. price range). In other words what are the 10 top variables (nlargest(10)) that will have an impact on the price range.**

**Type the following:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import ExtraTreesClassifier

#read in dataset

data = pd.read\_csv("mobile.csv")

X = data.iloc[:,0:20] #all rows for independent columns from Column 0 to 19

y = data.iloc[:, 20] #all rows for target column i.e price range (Column 20)

#Build the model

model = ExtraTreesClassifier()

model.fit(X,y)

#Use inbuilt feature\_importances of tree-based classifiers

print(model.feature\_importances\_)

#plot graph of feature importances for better visualization

feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(10).plot(kind='barh')

plt.show()

**Example 1.3.3 Choose all variables (nlargest(19) - because there are 19 independent variables) for the target variable (I.e. price range) and you manually decide which independent variables to choose as input variables.**

**Type the following:**

#Print feature importance for all

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import ExtraTreesClassifier

#read in dataset

data = pd.read\_csv("mobile.csv")

X = data.iloc[:,0:20] #independent columns

y = data.iloc[:, 20] #target column i.e price range

#Build the model

model = ExtraTreesClassifier()

model.fit(X,y)

#Use inbuilt feature\_importances of tree-based classifiers

print(model.feature\_importances\_)

#plot graph of feature importances for better visualization

feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(19).plot(kind='barh')

plt.show()

**Resource**

<https://machinelearningmastery.com/calculate-feature-importance-with-python/>

<https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=You%20can%20get%20the%20feature,feature%20towards%20your%20output%20variable>

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iloc.html>

<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

<https://seaborn.pydata.org/examples/many_pairwise_correlations.html>

**1.4 Correlation Coefficient and Hypotheses Testing**

The aim of the hypothesis testing is whether there is any linear relationship between two variables.

1. You will have to choose the confidence level (in this example, we shall choose confidence level as 95%, a= 0.05)
2. Assumptions – assumptions for a parametric test
3. Hypotheses
   1. H0: the two samples are independent (or correlation coefficient, r = 0)
   2. H1: there is a dependency between the samples (or correlation coefficient, r ≠ 0
4. Find the r and p values
5. Compare p value with a/2 = 0.025, and draw a conclusion

**Example 1.4.1 - Correlation Coefficient Testing for 2 number lists.**

**Type the following:**

#Correlation Hypothesis Testing by using pvalues

import pandas as pd

from scipy import stats

#Read data and build a dataframe

data = {'list 1':[2,4,6,8],'list 2':[4,16,36,64]}

df = pd.DataFrame(data, columns = ['list 1','list 2'])

#Calculate Pearson Correlation r and p value

pearsonCoeff\_rvalue, p\_value = stats.pearsonr(df["list 1"], df["list 2"]) #define the columns to perform calculations on

print("Pearson Correlation Coefficient r value : ", pearsonCoeff\_rvalue.round(decimals=3), "and a P-value of:", p\_value.round(decimals =3)) # Results

print("\n")

#Conduct Correlation Coefficient Hypothesis Testing

#Use 2 tail test

#Confidence level is 95%, alpha is 0.05 and alpha/2 is 0.025

alpha = 0.05

alpha\_half = 0.025

if p\_value < alpha\_half: # null hypothesis: x comes from a normal distribution

print("Conclusion drawn: The null hypothesis can be rejected")

else:

print("Conclusion drawn: The null hypothesis is accepted")

**Interpretation**

The same as two-tail t-test.

**Example 1.4.2 - Correlation Coefficient Hypothesis Testing for 2 numeric columns in the mobile.csv file.**

**Type the following:**

**#Correlation Coefficient (r) and Hypothesis Testing by using pvalues**

**import pandas as pd**

**from scipy import stats**

**#Read in the dataset**

**dataset = pd.read\_csv('mobile.csv')**

**df = pd.DataFrame(dataset)**

**print (df.head())**

**#Only explore 2 columns: Battery Power and Clock Speed**

**df1 = df['battery\_power']**

**df2 = df['clock\_speed']**

**#We would like to explore the relationship between Column 0 (battery\_power) and Column 2(clock\_speed)**

**pearson\_coef\_rvalue, p\_value = stats.pearsonr(df1, df2) #define the columns to perform calculations on**

**print("Pearson Correlation Coefficient: ", pearson\_coef\_rvalue.round(decimals=3), "and a P-value of:", p\_value.round(decimals=3)) # Results**

**print("\n")**

**#Conduct Correlation Coefficient Hypothesis Testing**

**#Use 2 tail test**

**#Confidence level is 98%, alpha is 0.05 and alpha/2 is 0.01**

**alpha = 0.02**

**alpha\_half = 0.01**

**if p\_value < alpha\_half: # null hypothesis: x comes from a normal distribution**

**print("Conclusion drawn: The null hypothesis can be rejected")**

**else:**

**print("Conclusion drawn: The null hypothesis is accepted")**

**Interpretation**

The same as two-tail t-test.

**Example 1.4.3 Further guide for the interpretation of the correlation coefficient hypothesis test results.**

**Types the following:**

#https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/

# Example of the Pearson's Correlation test

import pandas as pd

from scipy.stats import pearsonr

data1 = [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869]

data2 = [0.353, 3.517, 0.125, -7.545, -0.555, -1.536, 3.350, -1.578, -3.537, -1.579]

df = pd.DataFrame(data, columns = ['data1','data2'])

#Conduct Correlation Statistics

rvalue, pvalue = pearsonr(data1, data2)

print('stat=%.3f, p=%.3f' % (rvalue, pvalue))

print("\n")

#Confedence level is 99%, alpha = 0.01

alpha\_half = 0.005

if pvalue > 0.005:

print('Accept the null hypothesis: the conclusions are:')

print('(i) correlation coefficient, r value, is equal to 0;')

print('(ii) both variables are independent of each other;')

print('(iii) both variables do not have any relationship')

else:

print('Reject the null hypothesis: the conclusions are:')

print('(i) correlation coefficient, r value, is significantly different from 0;')

print('(ii) both variables are dependent of each other;')

print('(iii) both variables have a relationship')

**Spearman Rank Correlation**

**Note this is a non-parametric test. Notes are found here** <https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/>

**Sources**

[**https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pearsonr.html**](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pearsonr.html)

[**https://stackoverflow.com/questions/25571882/pandas-columns-correlation-with-statistical-significance**](https://stackoverflow.com/questions/25571882/pandas-columns-correlation-with-statistical-significance)

**Why hypothesis testing for correlation coefficient?**

[**https://courses.lumenlearning.com/boundless-statistics/chapter/hypothesis-testing-correlations/#:~:text=Testing%20the%20Significance%20of%20the,points%20are%20in%20the%20sample**](https://courses.lumenlearning.com/boundless-statistics/chapter/hypothesis-testing-correlations/#:~:text=Testing%20the%20Significance%20of%20the,points%20are%20in%20the%20sample)**.**

[**https://courses.lumenlearning.com/introstats1/chapter/testing-the-significance-of-the-correlation-coefficient/**](https://courses.lumenlearning.com/introstats1/chapter/testing-the-significance-of-the-correlation-coefficient/)

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html>

<https://realpython.com/numpy-scipy-pandas-correlation-python/>

<https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.pearsonr.html>

<https://medium.com/analytics-vidhya/basic-of-correlations-and-using-pandas-and-scipy-for-calculating-correlations-2d16c2bd6af0>

<https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/>

<https://www.kite.com/python/answers/how-to-find-the-correlation-between-two-pandas-dataframe-columns-in-python>

<https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html>

**2.Chi Square**

**2.1 Chi**-**square Goodness of Fit** **Test**

If there is only 1 categorical variable, then use Chi-square goodness of fit. test is a non-parametric (distribution-free) method used to compare the observed and expected values from one categorical variable.  Use scipy.stats.chisquare

**Example 2.1.1:** We assume to have two sets of data: (i) set of observed frequency for daily number of tourists that visit a museum in a week (f\_obs); (ii) set of expected frequency for daily number of tourists that visit the museum in a week (f\_exp).

The procedures for the Chi-Square Goodness of Fit Test are:

1. Choose a level of confidence. In this example, we shall choose 98%, a= 0.02
2. The hypotheses are as follows:

Null Hypothesis, Ho is there is no significant difference between the observed and expected values

Alternative Hypothesis, Ha is there is a significant difference between the observed and expected values

1. Interpret the results and draw a conclusion

**Type the following:**

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

#Chi Square Goodness Fit Test

from scipy.stats import chisquare

#Data for observed and expected frequencies from Monday to Sunday

f\_obs = [110, 120, 232, 150, 170, 100, 300]

f\_exp = [200, 200, 200, 200, 200, 250, 250]

#apply chisquare. ddof and axis is assumed to be 0.

chi\_sqValue, pvalue = chisquare(f\_obs, f\_exp)

print("Chi Square value is: ", chi\_sqValue)

print("\n")

print("P value is: ", pvalue)

print("\n")

The output is:

Chi Square value is: 194.62

P value is: 2.648587320540443e-39

**Interpretation of the Results**

The X2 value is 194.62 and p value, 2.648587320540443e-39 < a= 0.02.

Therefore, reject the null hypothesis, i.e. there is a significant difference between the observed and expected values.

**Sources**

<https://www.kite.com/python/examples/709/scipy-compute-the-chi-square-of-a-dataset>

**2.2 Chi Square Test of Independence**

Two categorical variables (independent variable) and a dependent variable.

The aim is to explore whether there is any dependence of the two categorical variables.

Interpretation of Chi Square Test of Independence

The Chi-square test of association evaluates relationships between categorical variables. Like any [statistical hypothesis test](https://statisticsbyjim.com/hypothesis-testing/hypothesis-tests-significance-levels-alpha-p-values/), the Chi-square test has both a [null hypothesis](https://statisticsbyjim.com/glossary/null-hypothesis/) and an [alternative hypothesis](https://statisticsbyjim.com/glossary/alternative-hypothesis/).

* [Null hypothesis](https://statisticsbyjim.com/glossary/null-hypothesis/): There are no relationships between the categorical variables. If you know the value of one variable, it does not help you predict the value of another variable.
* [Alternative hypothesis](https://statisticsbyjim.com/glossary/alternative-hypothesis/): There are relationships between the categorical variables. Knowing the value of one variable does help you predict the value of another variable.

**Example 2.2.1 We are going to use the titanic.txt dataset. Explanation notes are embedded as comments in the codes**

**Type the following:**

#Crosstab, Contingency Table and Chi Square

import pandas as pd

import scipy.stats as stats

#read in the titanic.txt file

titanic = pd.read\_csv("titanic.txt")

print(titanic.info())

print("\n")

print(titanic.describe())

print("\n")

print("Total number of males and females who survived in the dataset")

groupby\_sex = titanic.groupby(["Sex", "Survived"]).size()

print(groupby\_sex)

print("\n")

crosstab = pd.crosstab(titanic["Sex"], titanic["Survived"])

print(crosstab)

print("\n")

#Conduct the Chi Square Independence Test

#Interpretation https://www.pythonfordatascience.org/chi-square-test-of-independence-python/

#The information is returned within a tuple where the first value is the test static,

#the second value is the p-value,

#and the third number is the degrees of freedom.

#An array is also returned which contains the expected cell counts.

w, x, y, z = stats.chi2\_contingency(crosstab)

print("The Chi Square value is:", w)

print("The pvalue is:", x)

print("The value for degree of freedom is :", y)

print("Expected cell counts is:", z)

print("\n")

#To capture the relevant frequency values and store in variables

print("Total number of males and females who survived in the dataset")

a,b,c,d = titanic.groupby(["Sex", "Survived"]).size()

print("Number of females who died was:", a)

print("Number of females who survived was:", b)

print("Number of males who died was:", c)

print("Number of males who survived was:", d)

#Repeat the Chi Square Independence Test for the following table

print("\n")

print("Find the frequency count for grouped by sex (female and male), and cabin class (Class 1, 2, 3)")

groupby\_sex2 = titanic.groupby(["Sex", "Pclass"]).size()

crosstab2 = pd.crosstab(titanic["Sex"], titanic["Pclass"])

print(crosstab2)

print("\n")

k, l, m, n = stats.chi2\_contingency(crosstab2)

print("The Chi Square value is:", k)

print("The pvalue is:", l)

print("The value for degree of freedom is :", m)

print("Expected cell counts is:", n)

print("\n")

#Repeat the Chi Square Independence Test for the following table

print("\n")

print("Find the frequency count for grouped by survived (0,1), and cabin class (Class 1, 2, 3)")

groupby\_sex3 = titanic.groupby(["Survived", "Pclass"]).size()

crosstab3 = pd.crosstab(titanic["Survived"], titanic["Pclass"])

print(crosstab3)

print("\n")

f, g, h, i = stats.chi2\_contingency(crosstab3)

print("The Chi Square value is:", f)

print("The pvalue is:", g)

print("The value for degree of freedom is :", h)

print("Expected cell counts is:", i)

print("\n")

#Repeat the Chi Square Independence Test for the following table

print("\n")

print("Find the frequency count for grouped by sex (male, female), survived (0,1), and cabin class (Class 1, 2, 3)")

crosstab4 = pd.crosstab(titanic["Sex"], [titanic["Survived"], titanic["Pclass"]])

print(crosstab4)

print("\n")

p, q, r, s = stats.chi2\_contingency(crosstab4)

print("The Chi Square value is:", p)

print("The pvalue is:", q)

print("The value for degree of freedom is :", r)

print("Expected cell counts is:", s)

print("\n")

**Sources**

<https://statisticsbyjim.com/hypothesis-testing/chi-square-test-independence-example/#:~:text=For%20a%20Chi%2Dsquare%20test,exists%20between%20the%20categorical%20variables>.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html>

<https://www.pythonfordatascience.org/chi-square-test-of-independence-python/>

<https://www.pythonfordatascience.org/chi-square-test-of-independence-python/>

<https://pythonfordatascienceorg.wordpress.com/chi-square-python/>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html>

<https://pythonfordatascienceorg.wordpress.com/chi-square-python/>

<https://machinelearningmastery.com/chi-squared-test-for-machine-learning/>

<https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=You%20can%20get%20the%20feature,feature%20towards%20your%20output%20variable>.

<https://reneshbedre.github.io/blog/chisq.html>

<https://www.mathsisfun.com/data/chi-square-test.html>

<https://www.statisticshowto.com/probability-and-statistics/chi-square/>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html>

<https://machinelearningmastery.com/chi-squared-test-for-machine-learning/>

<https://www.kaggle.com/kuldeepnpatel/chi-square-test-of-independence>

**3.Non-Parametric Tests**

Comprehensive list of hypotheses testing

[**https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/**](https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/)

**Non-Parametric Tests (From Week 3)**

**Example 3.1 Normality Tests**

* + **Shapiro-Wilks Test for normal distribution**

The Shapiro-Wilk test tests the null hypothesis that the data was drawn from a normal distribution.

We shall investigate the ‘age’ column of the pima-indians-diabetes.xlsx file – whether it is drawn from a normal distribution and confidence level is 95%.

**Type the following:**

#Shapiro-Wilk Test

import pandas as pd

from scipy import stats

#read in the pima-indians-diabetese.xlsx dataset

dataset = pd.read\_excel('pima-indians-diabetes.xlsx')

df = pd.DataFrame(dataset, columns= ['pregNo', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age'])

print (df.head())

shapiro\_test = stats.shapiro(df['age'])

print("\n")

print('The Overall Shapiro-Wilk Test Result is: ', shapiro\_test)

print("\n")

print("The Shapiro Test Statistics value is: ",shapiro\_test.statistic)

print("\n")

print("The Shapiro Test pvalue is: ",shapiro\_test.pvalue)

**The Output**

The Overall Shapiro-Wilk Test Result is: ShapiroResult(statistic=0.8747662305831909, pvalue=2.401946561183188e-24)

The Shapiro Test Statistics value is: 0.8747662305831909

The Shapiro Test pvalue is: 2.401946561183188e-24

**The Interpretation**

The pvalue of 2.401946561183188e-24 < a= 0.02, therefor reject the null hypothesis that age is drawn from a normal distribution.

**Resources**

* + [**https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html**](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html)

* + **Quantile-Quantile (QQ) Plot for Normality**

We shall investigate the ‘age’ column of the pima-indians-diabetes.xlsx file.

**Type the following:**

import pandas as pd

from statsmodels.graphics.gofplots import qqplot

from matplotlib import pyplot as plt

#Load the dataset

dataset = pd.read\_excel('pima-indians-diabetes.xlsx')

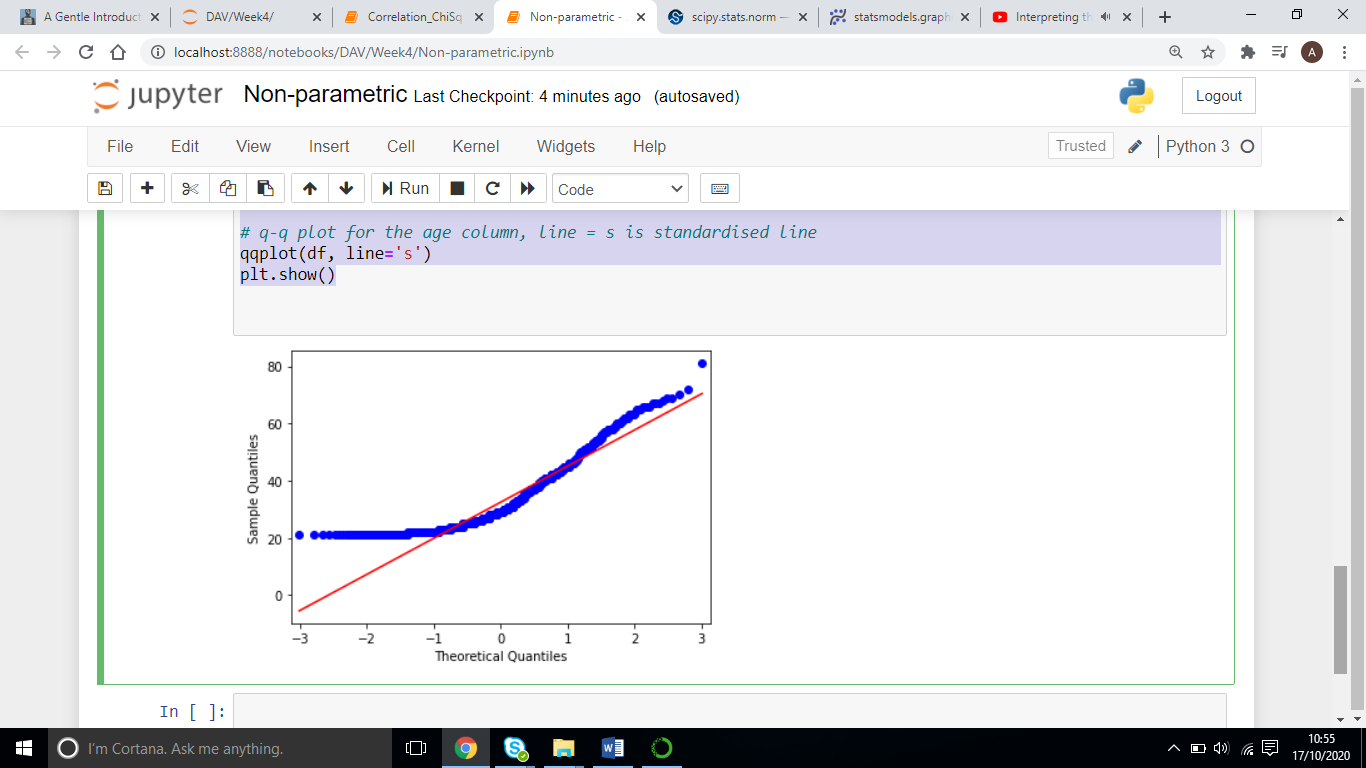
df = dataset['age']

# q-q plot for the age column, line = s is standardised line

qqplot(df, line='s')

plt.show()

**The output**

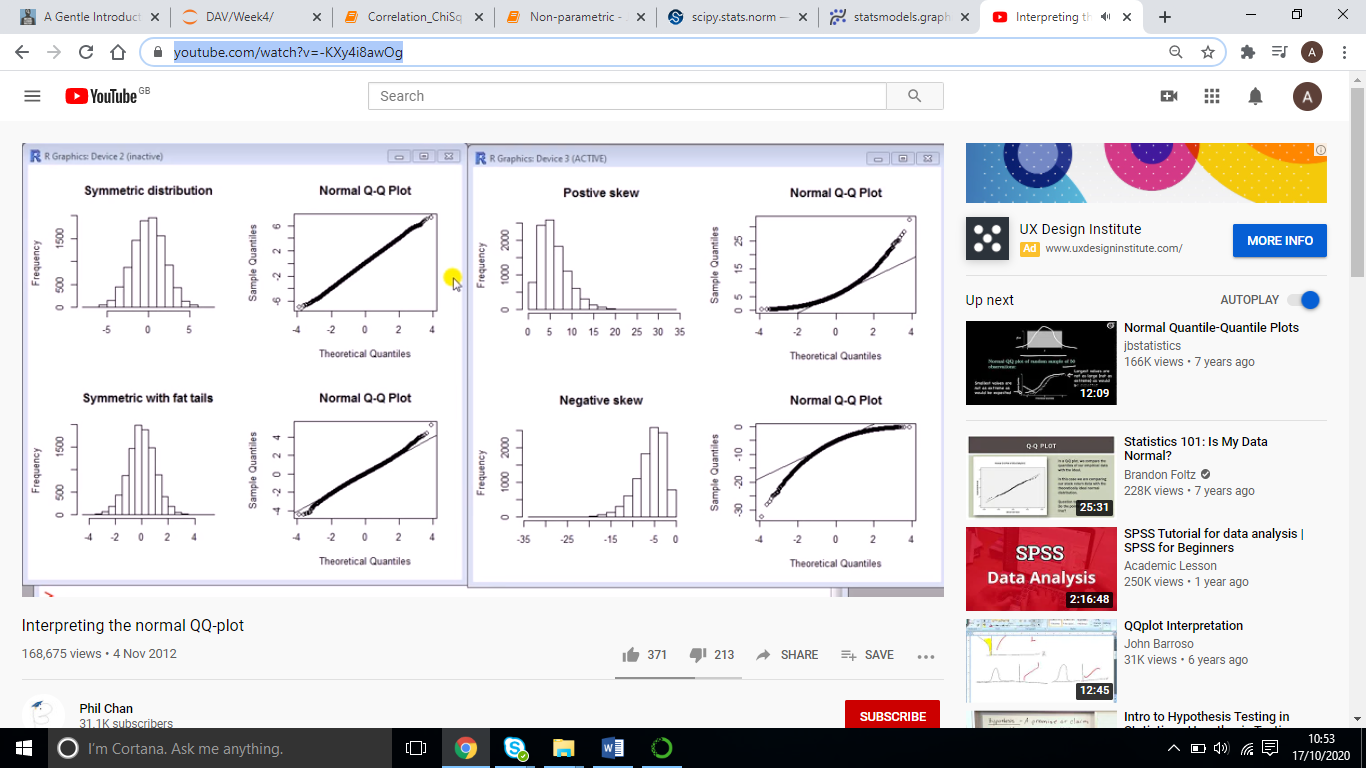


Interpretation

Compare with the qqplots below.

Which one does it have the best match?

https://www.youtube.com/watch?v=-KXy4i8awOg



**Resources**

<https://www.statsmodels.org/stable/generated/statsmodels.graphics.gofplots.qqplot.html>

Understanding qqplots <https://data.library.virginia.edu/understanding-q-q-plots/>

<https://www.youtube.com/watch?v=-KXy4i8awOg>

* + **Anderson-Darling test for normality**

Do on your own <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.anderson.html>

**Example 3.2 Homogeneity of variances when data is not drawn from normal distribution**

* + **Levene Test for Equal Variances**

Levene's test is a statistical procedure for testing equality of variances (also sometimes called homoscedasticity or homogeneity of variances) between two or more samples. Perform Levene test for equal variances.

The Levene test tests the null hypothesis that all input samples are from populations with equal variances.

median : Recommended for skewed (non-normal) distributions

mean : Recommended for symmetric, moderate-tailed distributions.

trimmed : Recommended for heavy-tailed distributions.

**The hypotheses testing is as follows:**

Ho : All of the k samples have equal variances.

Ha: At least one of the k sample variances is not equal.

**Example 3.2.1 Levene test for 3 number lists.**

**Type the following:**

from scipy.stats import levene

a = [8.88, 9.12, 9.04, 8.98, 9.00, 9.08, 9.01, 8.85, 9.06, 8.99]

b = [8.88, 8.95, 9.29, 9.44, 9.15, 9.58, 8.36, 9.18, 8.67, 9.05]

c = [8.95, 9.12, 8.95, 8.85, 9.03, 8.84, 9.07, 8.98, 8.86, 8.98]

stat, p = levene(a, b, c)

p

Output

P value is 0.002431505967249681

**Example 3.2.2 Conduct Levene Test for 3 columns in a dataset.**

**Type the following**

import pandas as pd

from scipy import stats

from scipy.stats import levene

import numpy as np

#read in the student marks file

dataset = pd.read\_excel("student\_marks.xlsx")

print(dataset.head())

#Obtain the marks for Subjects 1, 2, 3

S1 = dataset['Subject1']

S2 = dataset['Subject2']

S3 = dataset['Subject3']

#Conduct Levene Test for Subjects 1, 2, 3

stat1, p1 = levene(S1, S2, S3, center = "mean")

print("Test using mean ",p1)

stat2, p2 = levene(S1, S2, S3, center = "median")

print("Test using median ", p2)

Output and Interpretation

Pvalues

Test using mean 0.8537498129240625

Test using median 0.8600494878220108

**Example 3.2.3 Conduct Levene Test for subsets of dataset.**

**Type the following:**

import pandas as pd

from scipy import stats

from scipy.stats import levene

import numpy as np

#read in the student marks file

dataset = pd.read\_excel("student\_marks.xlsx")

print(dataset.head())

#Obtain the marks for Subjects 1, 2, 3

S1 = dataset['Subject1']

S2 = dataset['Subject2']

S3 = dataset['Subject3']

#Conduct Levene Test for Subjects 1, 2, 3

stat1, p1 = levene(S1, S2, S3, center = "mean")

print("Test using mean ",p1)

stat2, p2 = levene(S1, S2, S3, center = "median")

print("Test using median ", p2)

Ga, Gb, Gc = dataset.groupby(['Group'])['Subject1']

print(Ga)

print(Gb)

print(Gc)

groupA = dataset[dataset['Group'] == 'A']

print('Group A', groupA)

print("\n")

groupB = dataset[dataset['Group'] == 'B']

print('Group B', groupB)

print("\n")

groupC = dataset[dataset['Group'] == 'C']

print('Group C', groupC)

print("\n")

#Have to convert to List for Subject1

stat3, p3 = levene(list(groupA['Subject1']), list(groupB['Subject1']), list(groupC['Subject1']), center='mean')

print("Test using mean for Subject 1 for groups A, B, and C. The p value is: ",p3)

**Resources**

<https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.levene.html>

**Example 3.3 Mann Whitney is the non parametric alternative to the two sample t test**

**Example 3.3.1 Conduct Mann Whitney test for two number lists.**

**Type the following:**

import numpy as np

import scipy.stats as stats

# Create two groups of data

group1 = [1, 5 ,7 ,3 ,5 ,8 ,34 ,1 ,3 ,5 ,200, 3]

group2 = [10, 18, 11, 12, 15, 19, 9, 17, 1, 22, 9, 8]

# Calculate u and probability of a difference

u\_statistic, pVal = stats.mannwhitneyu(group1, group2)

# Print results

print ('P value:')

print (pVal)

**Interpretation:**

Similar to one-tail or two-tail t-tests.

**Example 3.3.2 Conduct Mann Whitney test for mean ages of salary > 50K and salary <= 50K. Type the following:**

#Source

#https://techrando.com/2019/11/21/determine-if-two-distributions-are-significantly-different-using-the-mann-whitney-u-test/

import pandas as pd

import numpy as np

from scipy import stats

df = pd.read\_csv('adult.csv', header = None)

#Declare the column names of the data set

df.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num',

'marital-status', 'occupation', 'relationship',

'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week',

'native-country', 'salary']

df\_less\_than\_50k = df[df['salary'] == ' <=50K']

df\_greater\_than\_50k = df[df['salary'] == ' >50K']

u\_statistic, p\_value = stats.mannwhitneyu(list(df\_less\_than\_50k['age']), list(df\_greater\_than\_50k['age']))

print(u\_statistic, p\_value)

**Interpretation:**

Similar to one-tail or two-tail t-tests.

**Sources**

<https://scipy-lectures.org/packages/statistics/index.html#the-pandas-data-frame>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html>

[**https://techrando.com/2019/11/21/determine-if-two-distributions-are-significantly-different-using-the-mann-whitney-u-test/**](https://techrando.com/2019/11/21/determine-if-two-distributions-are-significantly-different-using-the-mann-whitney-u-test/)

1. **Wilcoxon test is a nonparametric statistical test that compares two paired groups**

The Wilcoxon signed-rank test tests the null hypothesis that two related paired samples come from the same distribution. In particular, it tests whether the distribution of the differences x - y is symmetric about zero. It is a non-parametric version of the paired T-test.

<https://scipy-lectures.org/packages/statistics/index.html#the-pandas-data-frame>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html>

1. **Kruskal Willis test is the non parametric alternative to the One way ANOVA**

The Kruskal-Wallis H-test tests the null hypothesis that the population median of all of the groups are equal.

Resources

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kruskal.html>

1. **Friedman Repeated Tests**

The Friedman test tests the null hypothesis that repeated measurements of the same individuals have the same distribution. It is often used to test for consistency among measurements obtained in different ways.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kruskal.html#scipy.stats.kruskal>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.friedmanchisquare.html>

1. **Kolmogorov-Smirnov test for goodness of fit**

The KS test is only valid for continuous distributions. The two-sample test tests whether the two independent samples are drawn from the same continuous distribution.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html>

Datasets

<https://www.lock5stat.com/datapage.html>