Fernando Cerriteño Magaña A01702790

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
```

```
d={'jugo_A':[20, 25, 22, 23, 28, 26, 24, 21, 27, 25, 24, 22, 23, 26, 25, 23, 24, 22, 27, 26, 25, 24, 23, 22, 21, 26, 24, 25, 22, 23],
   'jugo_B':[19, 18, 21, 20, 23, 22, 20, 19, 22, 21, 20, 19, 18, 23, 22, 21, 20, 19, 23, 22, 21, 20, 19, 18, 23, 22, 21, 20, 19, 18]}
```

```
df = pd.DataFrame(data=d)
```

```
df.head()
```

	jugo_A	jugo_B
0	20	19
1	25	18
2	22	21
3	23	20
4	28	23

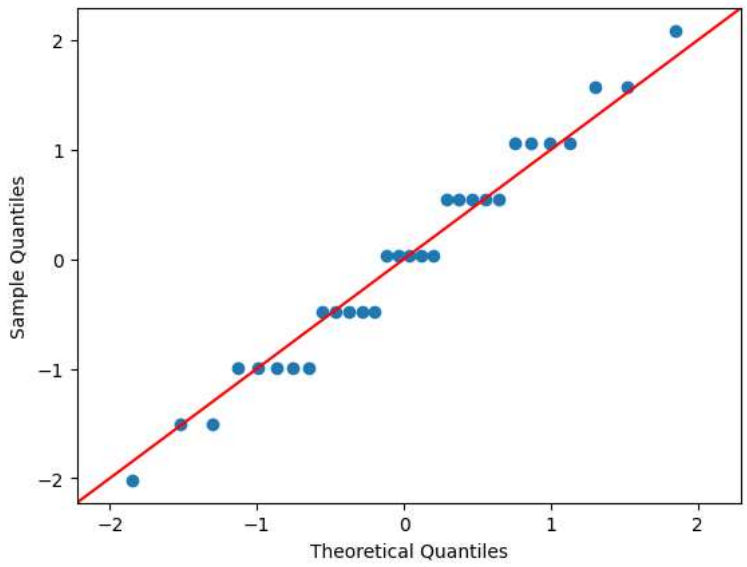
```
scaler = StandardScaler()
df_standard = scaler.fit_transform(df)
```

```
df_standard = pd.DataFrame(df_standard,columns=df.columns)
```

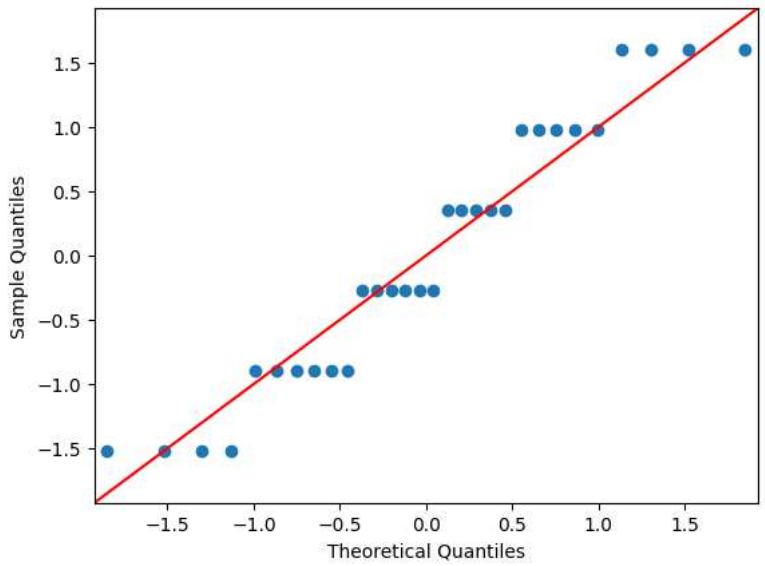
```
df_standard.head()
```

	jugo_A	jugo_B
0	-2.018938	-0.892547
1	0.547509	-1.515253
2	-0.992360	0.352867
3	-0.479070	-0.269840
4	2.087377	1.598281

```
figura_A = sm.qqplot(df_standard['jugo_A'],line="45")
```



```
figura_B = sm.qqplot(df_standard['jugo_B'],line="45")
```




```
statistic_A, p_value_A = stats.kstest(df_standard['jugo_A'], 'norm')

alpha = 0.05

if p_value_A < alpha:
    print('Los datos de la columna jugo_A no siguen una distribución normal')
else:
    print('Los datos de la columna jugo_A siguen una distribución normal')
```

```
 Los datos de la columna jugo_A siguen una distribución normal
```

```
statistic_B, p_value_B = stats.kstest(df_standard['jugo_B'], 'norm')

if p_value_B < alpha:
    print('Los datos de la columna jugo_B no siguen una distribución normal')
else:
    print('Los datos de la columna jugo_B siguen una distribución normal')
```

```
Los datos de la columna jugo_B siguen una distribución normal
```

Media y desviación estandar

```
meanA = np.mean(df['jugo_A'])
std_devA = np.std(df['jugo_A'], ddof=1)

meanB = np.mean(df['jugo_B'])
std_devB = np.std(df['jugo_B'], ddof=1)
```

```
Vc = 0.99
z = stats.norm.ppf((1 + Vc) / 2)
margen_ErrorA = z * (std_devA / np.sqrt(len(df['jugo_A'])))
margen_ErrorB = z * (std_devB / np.sqrt(len(df['jugo_B'])))

# Se calcula el intervalo de confianza
int_ConfA = (meanA - margen_ErrorA, meanA + margen_ErrorA)
int_ConfB = (meanB - margen_ErrorB, meanB + margen_ErrorB)

print("Intervalo de Confianza al 99%:", int_ConfA)
print("Intervalo de Confianza al 99%:", int_ConfB)
```

```
 Intervalo de Confianza al 99%: (23.00146241851692, 24.865204248149748)
 Intervalo de Confianza al 99%: (19.665203918875598, 21.20146274779107)
```