```python
# Bibliotecas básicas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Desviación estándar
# Conjunto ficticio con distribución normal, notar la 'n' al final
np.random.seed(0)
data = np.random.randn(50000) * 20 + 20
data[1], data[-1]
```

```
(np.float64(28.003144167344466), np.float64(-5.016539269717178))
```

```python
# Anomalías con std VERSIÓN PROPIA
corte = 3
std = np.std(data)
media = np.mean(data)
#              antes de 3 std             después de 3 std
anom = data[(data < media-corte*std) | (data > media+corte*std)]
len(anom)
```

```
148
```

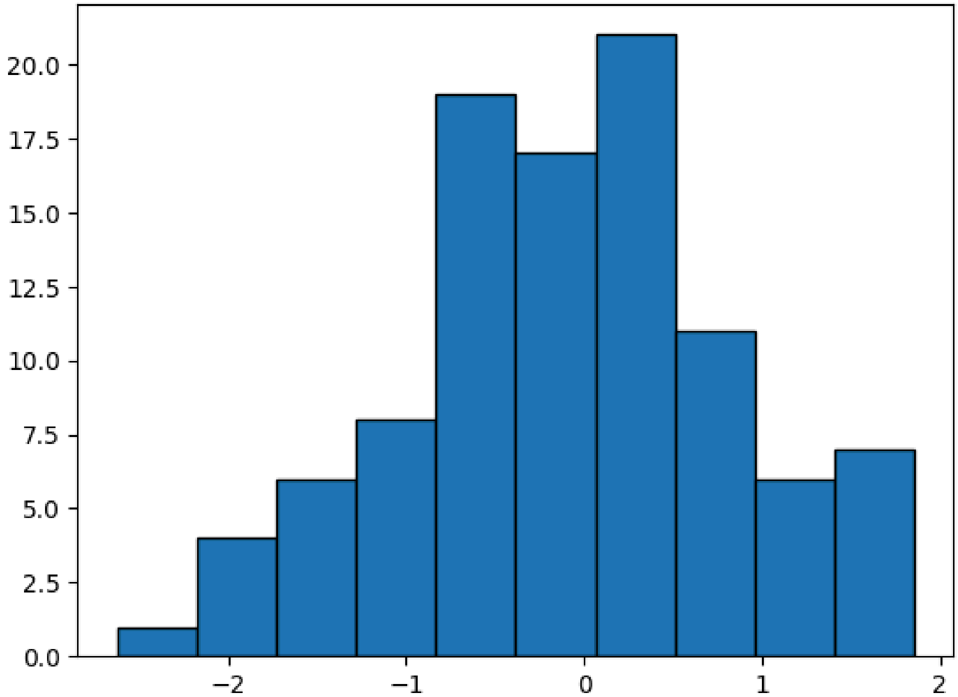
```python
# Anomalías con std VERSIÓN DE LALO
corte = 3
std = np.std(data)
media = np.mean(data)
#              antes de 3 std             después de 3 std
anom = data[(data < media-corte*std) | (data > media+corte*std)]
len(anom)
```

```
148
```

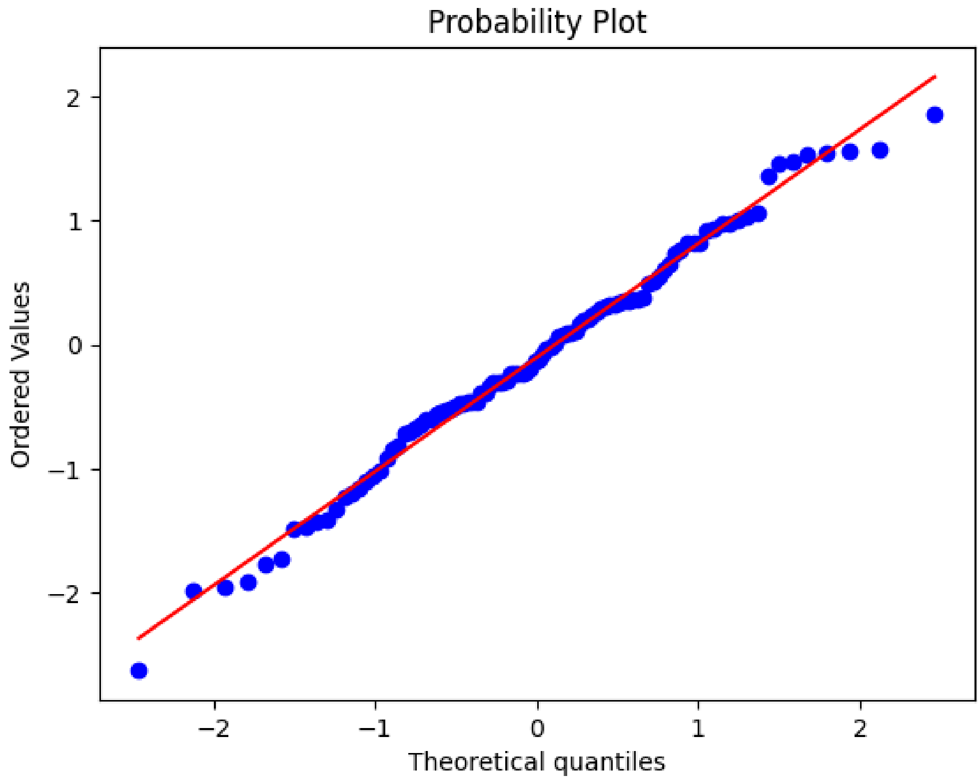
```python
# Prueba de normalidad: histograma, Quantile-quantile graph, Shapiro
# Conjunto de datos ficticio normal
np.random.seed(42)
data = np.random.normal(0,1,100)
```

```python
# 1. Histograma
plt.hist(data, edgecolor='black', linewidth=1)
```

```
(array([ 1.,  4.,  6.,  8., 19., 17., 21., 11.,  6.,  7.]),
 array([-2.6197451 , -2.17254278, -1.72534045, -1.27813812, -0.83093579,
        -0.38373346,  0.06346887,  0.5106712 ,  0.95787353,  1.40507586,
         1.85227818]),
 <BarContainer object of 10 artists>)
```




```
# 2. gráfica de cuantiles
import pylab
import scipy.stats as stats
prueba = stats.probplot(data, dist='norm', plot=pylab)
pylab.show()
```




```
# prueba
```

```
# 3. Prueba de Shapiro
from scipy.stats import shapiro
estad, p_value = shapiro(data)
print('p_value = ',p_value)
# p_value > 0.05 => distribución normal
```

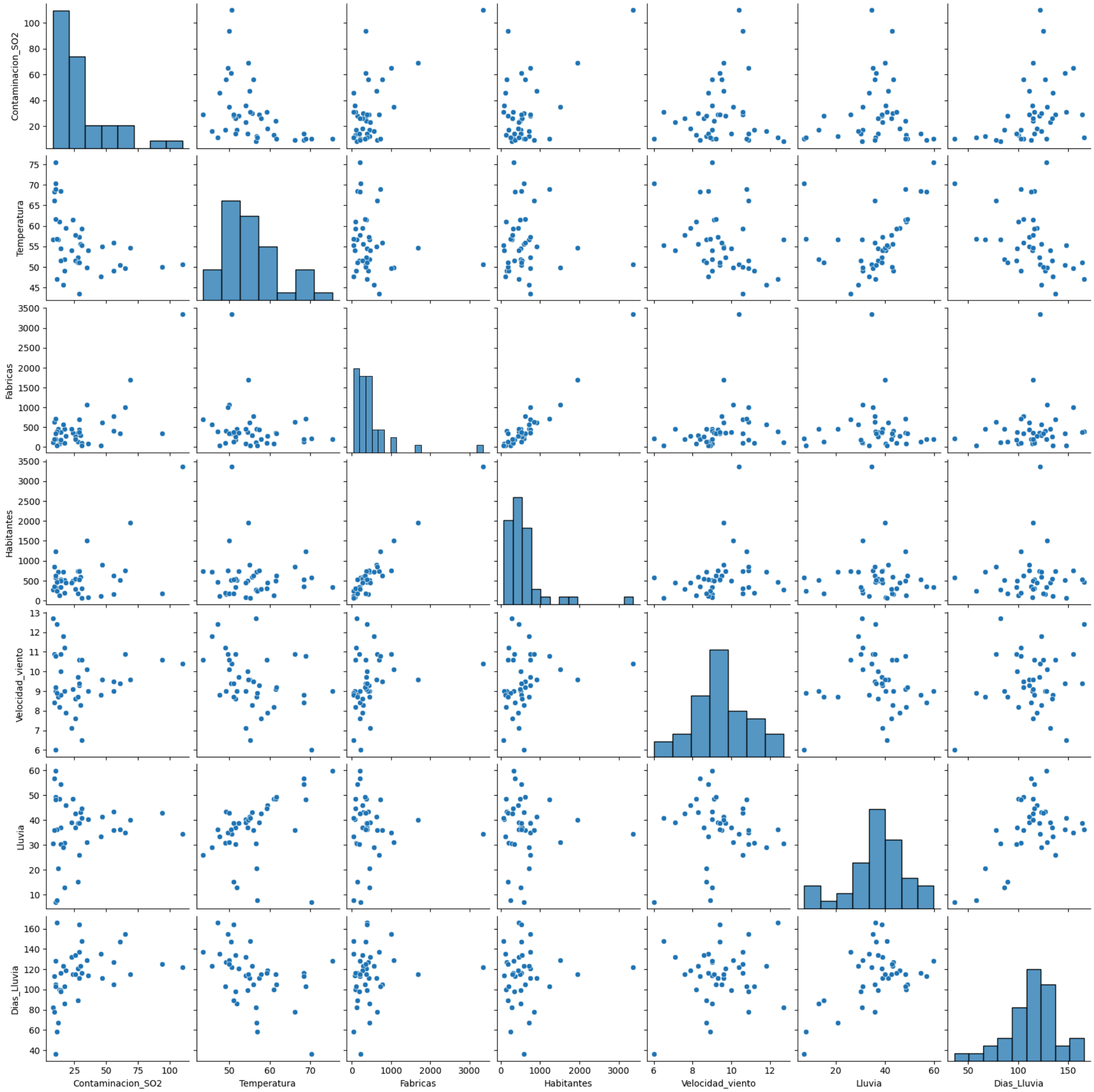
```
p_value =  0.6551676754214818
```

```
# Conjunto de datos real
cont = pd.read_csv('https://bit.ly/31B56KB')
cont.info()
```

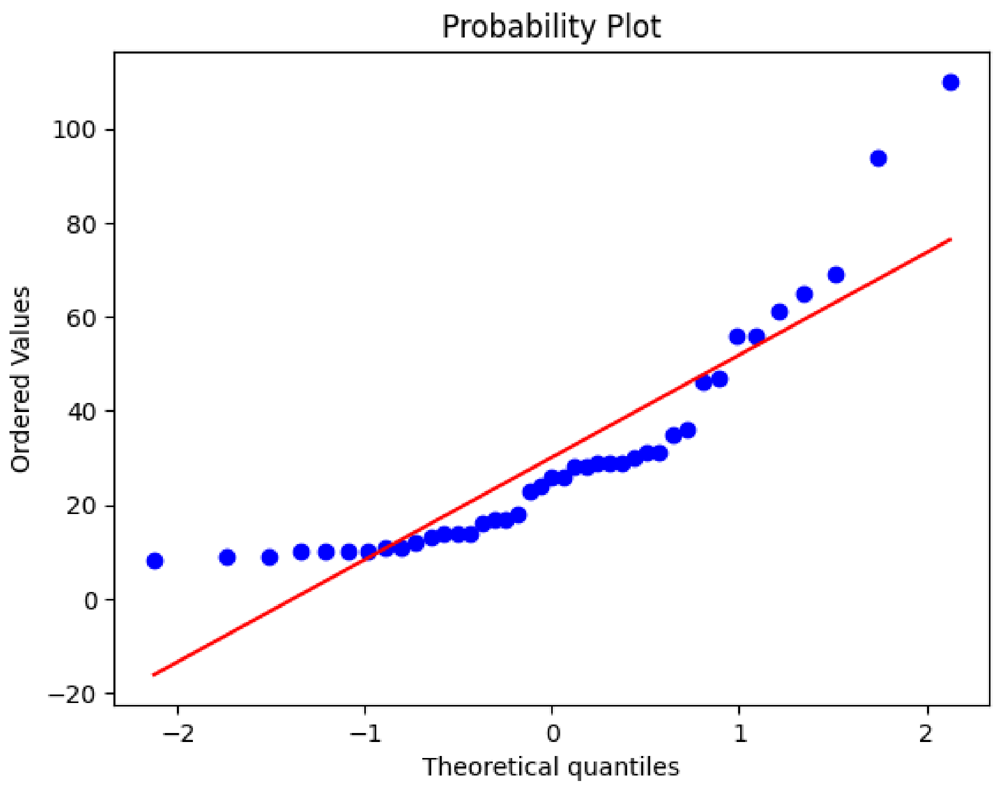
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41 entries, 0 to 40
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Contaminacion_SO2   41 non-null     int64  
 1   Temperatura         41 non-null     float64
 2   Fabricas            41 non-null     int64  
 3   Habitantes          41 non-null     int64  
 4   Velocidad_viento    41 non-null     float64
 5   Lluvia              41 non-null     float64
 6   Dias_Lluvia         41 non-null     int64  
dtypes: float64(3), int64(4)
memory usage: 2.4 KB
```

```
import seaborn as sns
sns.pairplot(cont)
```

<seaborn.axisgrid.PairGrid at 0x7f0ec084e590>



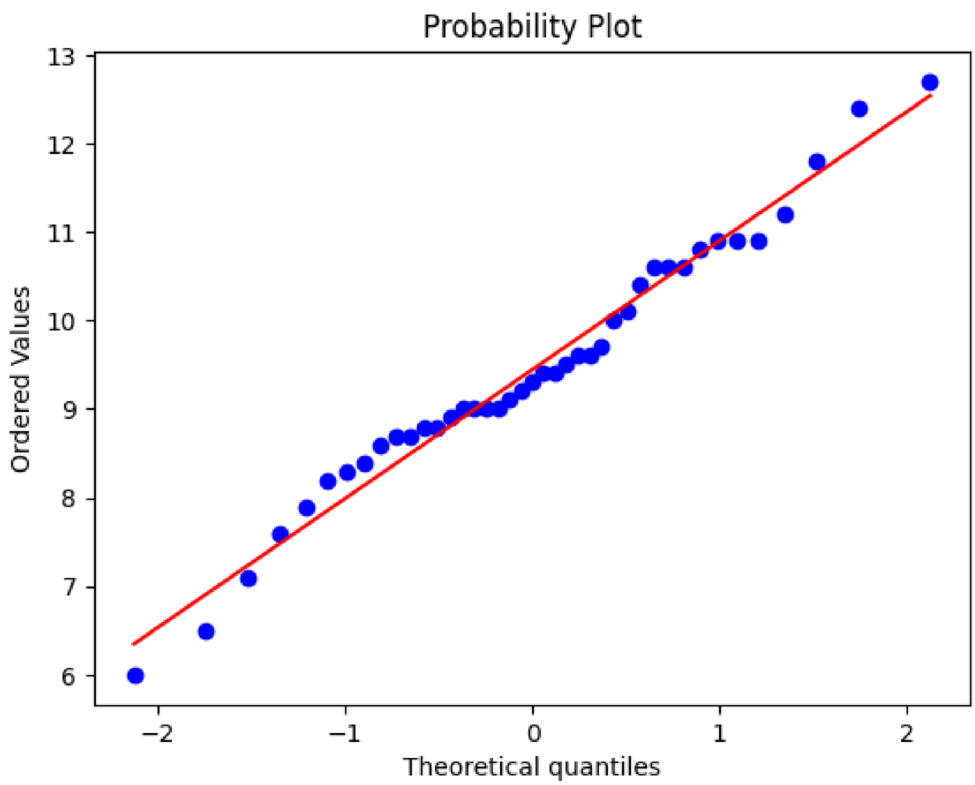

```
# Contaminación_SO2
# Quartiles
stats.probplot(cont.Contaminacion_SO2, dist='norm', plot=pylab)
pylab.show()
```




```
# Shapiro
estad, p_value = shapiro(cont.Contaminacion_SO2)
print('p_value = ', p_value)
```

```
p_value =  9.723376400158156e-06
```

```
# Velocidad_viento
stats.probplot(cont.Velocidad_viento, dist='norm', plot=pylab)
pylab.show()
```




```
# Shapiro
estad, p_value = shapiro(cont.Velocidad_viento)
print('p_value = ', p_value)
```

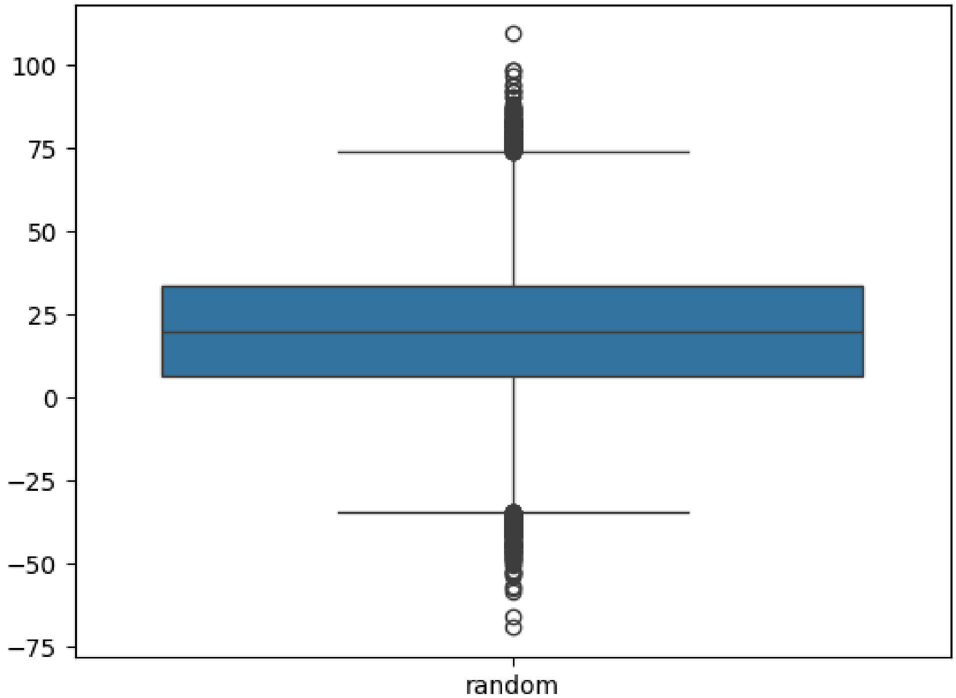
```
p_value =  0.6972579783041465
```

```
# Diagrama de bigotes para datos con distribución normal
data = np.random.randn(50000) * 20 + 20
data = pd.DataFrame(data, columns=['random'])
data.head(2)
```

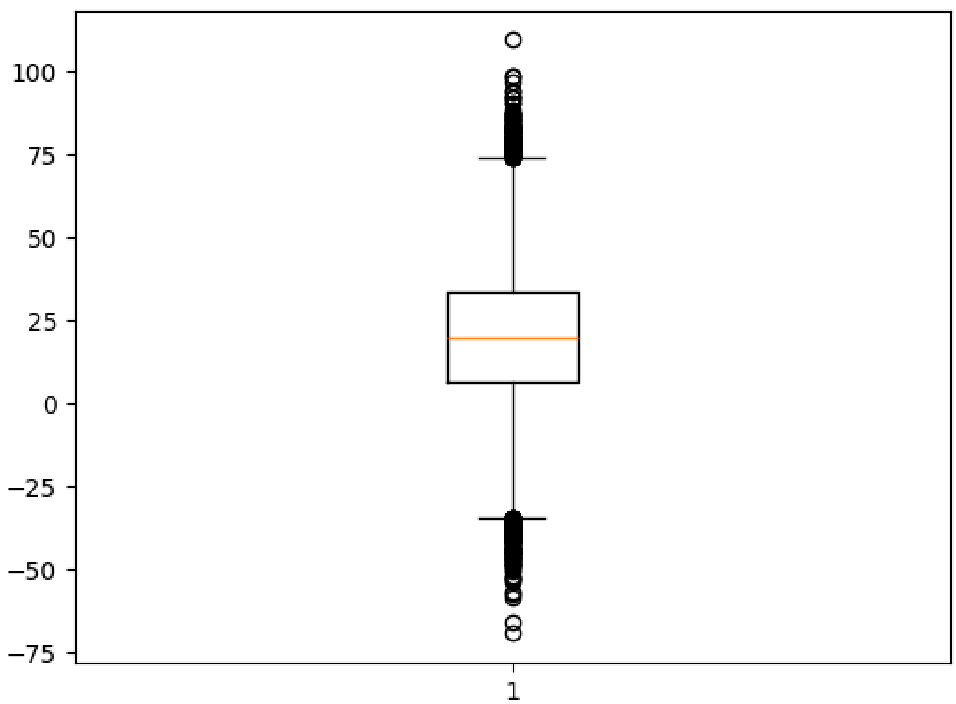
	random
0	-8.307415
1	11.587094

```
import seaborn as sns
sns.boxplot(data)
```

```
<Axes: >
```




```python
1 bp = plt.boxplot(x=data)
```




```python
1 bp
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f0ebdf7dad0>,
  <matplotlib.lines.Line2D at 0x7f0ebc28a410>],
 'caps': [<matplotlib.lines.Line2D at 0x7f0ebc23ff10>,
  <matplotlib.lines.Line2D at 0x7f0ebc23db90>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f0ebdf7e550>],
 'medians': [<matplotlib.lines.Line2D at 0x7f0ebc23dcd0>],
```

```
 'fliers': [<matplotlib.lines.Line2D at 0x7f0ebc23cfd0>],
 'means': []}
```

```
bp['fliers']
```

```
[<matplotlib.lines.Line2D at 0x7f0ebc23cfd0>]
```

```
#bp['fliers'][0].get_data()[1]
```

```
bp['fliers'][0].get_data()[1].shape
```

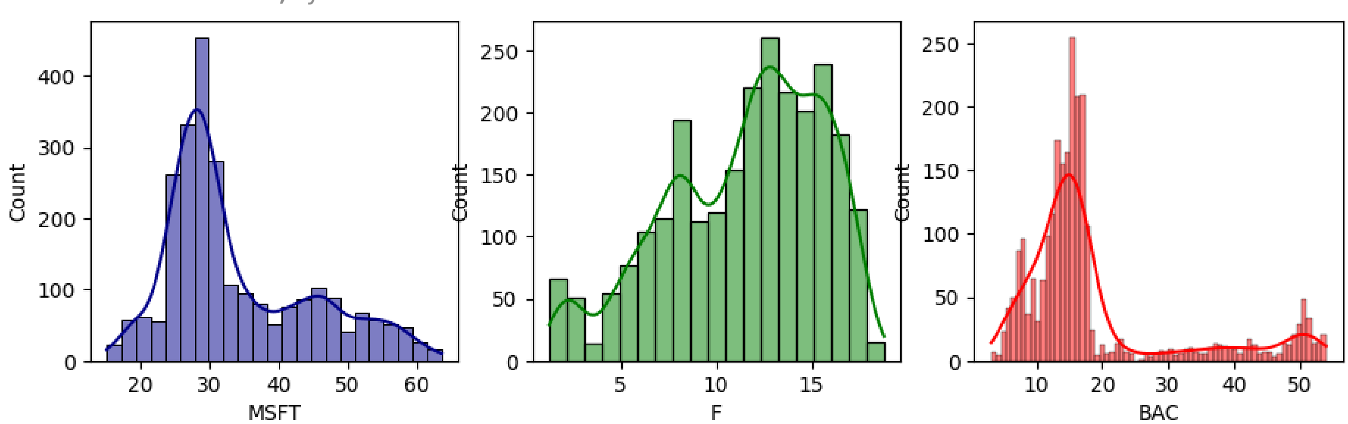
```
(332,)
```

```
stocks = pd.read_csv('stocks.csv', header='infer')
stocks.index = stocks.Date
#stocks.set_index(stocks.Date)
stocks.drop(['Date'],axis=1, inplace=True)
stocks.head(3)
```

	MSFT	F	BAC
Date			
1/3/2007	29.860001	7.51	53.330002
1/4/2007	29.809999	7.70	53.669998
1/5/2007	29.639999	7.62	53.240002

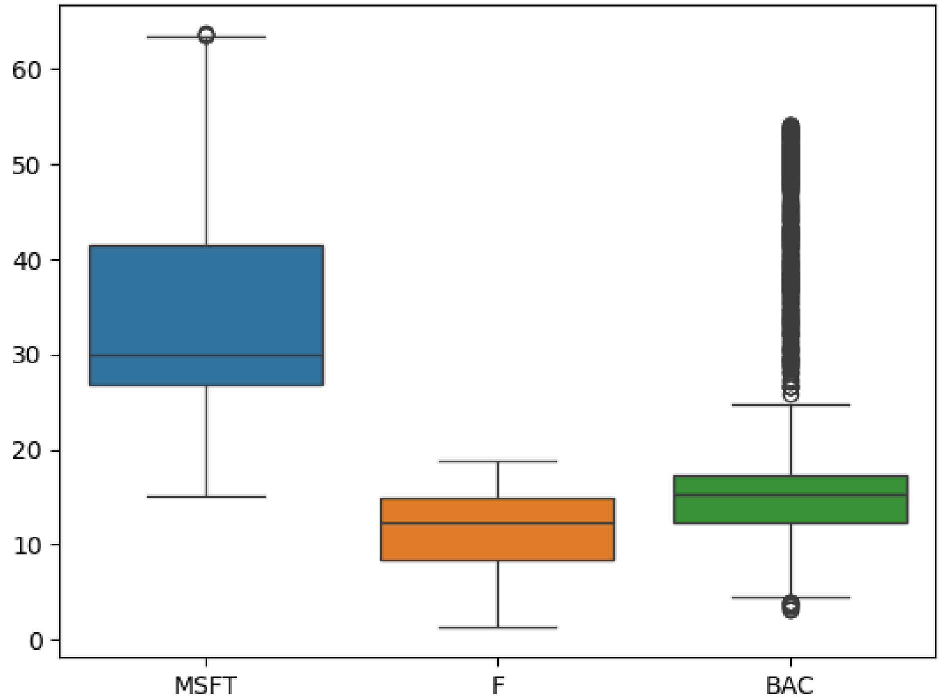
```
import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(11,3))
sns.histplot(stocks.MSFT, ax=ax[0], color="darkblue", kde=True)
sns.histplot(stocks.F, ax=ax[1], color='green', kde=True)
sns.histplot(stocks.BAC, ax=ax[2], color='red', kde=True)
```

```
<Axes: xlabel='BAC', ylabel='Count'>
```

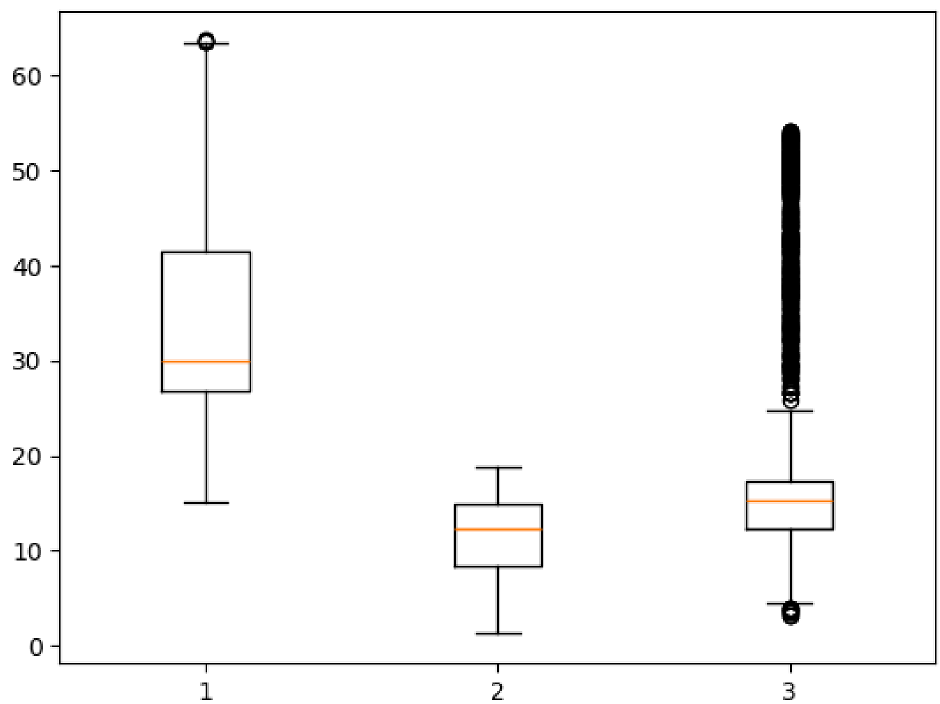



```
sns.boxplot(data=stocks)
```

<Axes: >

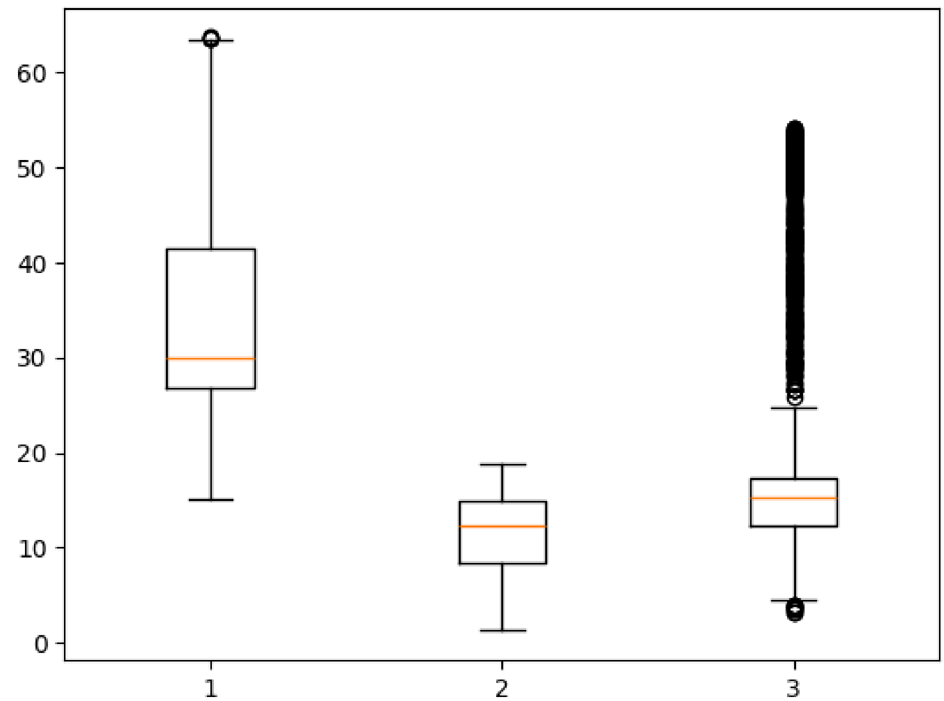



```
1 bp = plt.boxplot(x=stocks)
```




```
1 bp['fliers'][0].get_data()[1].shape
```

```
1 bp = plt.boxplot(x=stocks)
```




```
bp['fliers'][0].get_data()[1].shape
```

```
(4,)
```

```
bp['fliers'][1].get_data()[1].shape
```

```
(0,)
```

```
bp['fliers'][2].get_data()[1].shape
```

```
(441,)
```

```

```

```
# Obtener los atípicos con IQR
np.random.seed(0)
data = np.random.randn(50000) * 20 + 20
```

```
Q1, Q3 = np.percentile(data, [25, 75])
IQR = Q3-Q1
# bigotes (whiskers)
sup = Q3 + 1.5*IQR
inf = Q1 - 1.5*IQR
# anomalías
anom_ind = np.where((data < inf) | (data > sup))
anom = data[anom_ind] # Acceso directo a las anomalías
len(anom_ind[0]), len(anom)
```

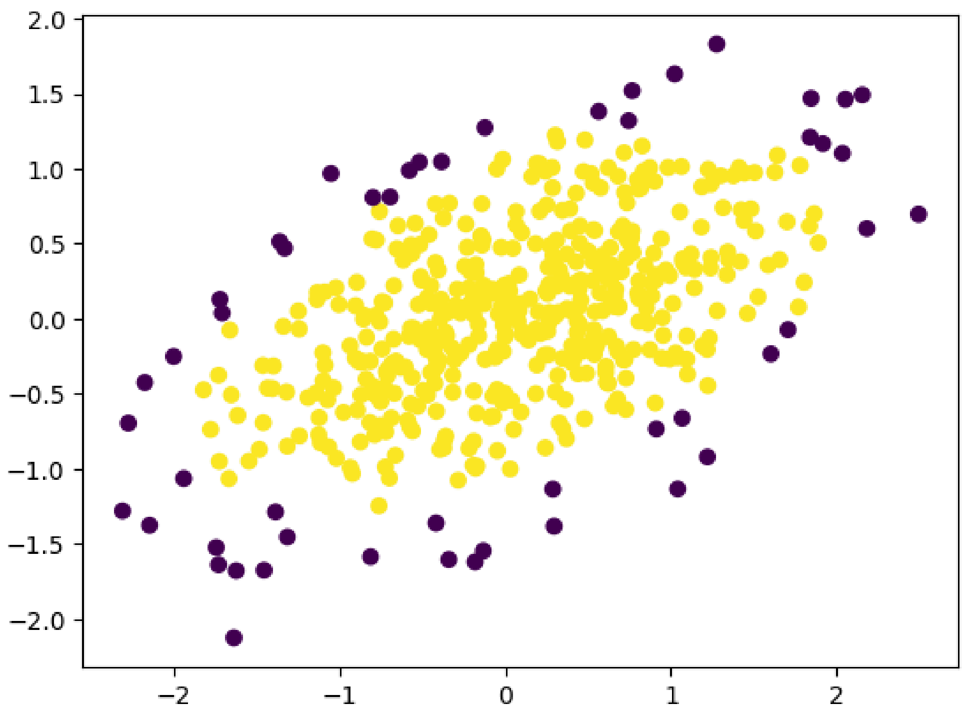
```
(379, 379)
```

```python
# EllipticEnvelope:
# Basado en covarianza, genera una elipse que "envuelve" los datos "normales"
# y deja fuera losa típicos
import numpy as np
import matplotlib.pyplot as plt
from sklearn.covariance import EllipticEnvelope
cov = np.array([[.8, .3],
                [.3, .4]])
X = np.random.RandomState(0).multivariate_normal(mean=[0,0],cov=cov,size=500)
ee = EllipticEnvelope(random_state=42) # random_state parámetro de replicabilidad
ee.fit(X)
# predict regresa 1 para valores normales y -1 para atípicos
anom = ee.predict(X)
ee.covariance_, ee.location_
```

```
(array([[0.7361743 , 0.25159568],
        [0.25159568, 0.30556494]]),
 array([0.07405583, 0.03661301]))
```

```python
plt.scatter(X[:,0],X[:,1],c=anom)
```

```
<matplotlib.collections.PathCollection at 0x7f0eb4471310>
```



```python
#anom
```

```python
# Funciona para datos con múltiples dimensiones
```